International Journal on Advances in Systems and Measurements

2013 vol. 6 nr. 3&4

The *International Journal on Advances in Systems and Measurements* is published by IARIA.
ISSN: 1942-261x
journals site: http://www.iariajournals.org
contact: petre@iaria.org

International Journal on Advances in Systems and Measurements
Volume 6, Number 3 & 4, 2013

Stefano Mariani, Politecnico di Milano, Italy
Paulo Martins Pedro, Chaminade University, USA / Unicamp, Brazil
Daisuke Mashima, Georgia Institute of Technology, USA
Don McNickle, University of Canterbury, New Zealand
Mahmoud Meribout, The Petroleum Institute - Abu Dhabi, UAE
Luca Mesin, Politecnico di Torino, Italy
Marco Mevius, HTWG Konstanz, Germany
Marek Miskowicz, AGH University of Science and Technology, Poland
Jean-Henry Morin, University of Geneva, Switzerland
Fabrice Mourlin, Paris 12th University, France
Adrian Muscat, University of Malta, Malta
Mahmuda Naznin, Bangladesh University of Engineering and Technology, Bangladesh
George Oikonomou, University of Bristol, UK
Arnaldo S. R. Oliveira, Universidade de Aveiro-DETI / Instituto de Telecomunicações, Portugal
Aida Omerovic, SINTEF ICT, Norway
Victor Ovchinnikov, Aalto University, Finland
Telhat Özdoğan, Recep Tayyip Erdogan University, Turkey
Gurkan Ozhan, Middle East Technical University, Turkey
Constantin Paleologu, University Politehnica of Bucharest, Romania
Matteo G A Paris, Universita` degli Studi di Milano,Italy
Vittorio M.N. Passaro, Politecnico di Bari, Italy
Giuseppe Patanè, CNR-IMATI, Italy
Marek Penhaker, VSB- Technical University of Ostrava, Czech Republic
Juho Perälä, VTT Technical Research Centre of Finland, Finland
Florian Pinel, T.J.Watson Research Center, IBM, USA
Ana-Catalina Plesa, German Aerospace Center, Germany
Miodrag Potkonjak, University of California - Los Angeles, USA
Alessandro Pozzebon, University of Siena, Italy
Vladimir Privman, Clarkson University, USA
Konandur Rajanna, Indian Institute of Science, India
Stefan Rass, Universität Klagenfurt, Austria
Candid Reig, University of Valencia, Spain
Teresa Restivo, University of Porto, Portugal
Leon Reznik, Rochester Institute of Technology, USA
Gerasimos Rigatos, Harper-Adams University College, UK
Luis Roa Oppliger, Universidad de Concepción, Chile
Ivan Rodero, Rutgers University - Piscataway, USA
Lorenzo Rubio Arjona, Universitat Politècnica de València, Spain
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Subhash Saini, NASA, USA
Mikko Sallinen, University of Oulu, Finland
Christian Schanes, Vienna University of Technology, Austria
Rainer Schönbein, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB), Germany
Guodong Shao, National Institute of Standards and Technology (NIST), USA
Dongwan Shin, New Mexico Tech, USA

International Journal on Advances in Systems and Measurements
Volume 6, Numbers 3 & 4, 2013

CONTENTS

Modeling of Expert Knowledge for Maritime Situation Assessment

Yvonne Fischer* and Jürgen Beyerer*†
*Vision and Fusion Laboratory, Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
†Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB)
Karlsruhe, Germany
Email: yvonne.fischer@kit.edu, juergen.beyerer@iosb.fraunhofer.de

***Abstract*—In today's surveillance systems, there is a need for enhancing the situation awareness of an operator. Supporting the situation assessment process can be done by extending the system with a module for automatic interpretation of the observed environment. In this article, the information flow in intelligent surveillance systems is described and a detailed modeling of the situation assessment process is presented. The main contribution of this article is a probabilistic modeling of situations of interest. The result of this modeling is a Situational Dependency Network (SDN), which represents the dependencies between several situations of different abstraction levels. The focus is on a top-down approach, i.e., the modeling is done in a human-understandable way and can be done by maritime experts. As especially critical situations can change very fast in their characteristics and also they do not happen very often, the machine learning approach is not appropriate for detecting such situations, even if they are very powerful. Therefore, we present an approach, where expert knowledge can be included into a Dynamic Bayesian Network (DBN). In this article, we will show how a DBN can be generated automatically from the SDN. We mainly focus on the determination of the parameters of the model, as this is the crucial point. The resulting DBN can then be applied to vessel tracks and the probability of the modeled situations of interest can be inferred over time. Finally, we present an example in the maritime domain and show that the probabilistic model yields the expected results.**

Keywords*—*surveillance system; situation awareness; situation assessment; data fusion; dynamic Bayesian networks; probabilistic reasoning.

I. Introduction

This article is based on our previous work that was presented at the ICONS 2012 conference and published in [1]. In this section, we will describe the motivation of our approach.

During the operation of complex systems that include human decision making, the processes of acquiring and interpreting information from the environment forms the basis for the state of knowledge of a decision maker. This mental state is often referred to as situation awareness [2], whereas the processes to achieve and maintain that state is referred to as situation assessment. In today's maritime surveillance systems, the situation assessment process is highly supported through various heterogeneous sensors and appropriate signal processing methods for extracting as much information as possible about the surveyed environment and its elements. They are equipped with powerful sensors like radar systems, the Automatic Identification System (AIS), see [3], or even infrared cameras. All of these sensor systems, either active or passive, are able to detect vessels in the observed area. The surveillance system fuses the estimated positions into consistent tracks, which then can be displayed in a dynamic map. Maritime surveillance systems have been used so far to monitor traffic, to guide passing vessels, and to ensure compliance with traffic regulations. Using these sensor systems is, of course, an essential capability for every surveillance system in order to be able to observe a designated area and to detect and track objects inside this area.

But there is an increasing demand for using their power in security-related applications like the detection of critical situations. However, current systems do not provide any support in the detection of spatio-temporal patterns, i.e., situations. Thus, there is a need for concepts and methods that are able to infer real situations from observed elements in the environment and to project their status in the near future. The challenge of intelligent surveillance systems is therefore not only to collect as much sensor data as possible, but also to detect and assess complex situations that evolve over time as an automatic support to an operator's situation assessment process, and therefore enhancing his situation awareness.

In applications like this, the current approach is to learn the characteristics, i.e., the features of the situation of interest with some training data, and recognizing the situations based on the trained model. There is a lot of machine learning methods that can be used for detecting sequential patterns, see for example [4] or [5]. All methods have in common that they need training data. In order to use such machine learning techniques for estimating maritime situations, there are several challenges that have to be addressed and we will highlight them in the following.

- *Data Collection:* There has to be an access to sensors that are able to collect vessel data. Data of different sensors, e.g., AIS and radar, should be fused into a consistent representation of vessel tracks. The representation should not only contain position data over time, but also additional information like speed, course, MMSI (maritime mobile service identity), beam, length, etc. Since surveillance systems are equipped with these sensors, data collection can be provided easily.
- *Data Labeling:* The collected data has to be labeled in order to perform supervised training. As we want to

detect specific situations, the vessel tracks have to be labeled, i.e., it has to be determined for which time interval the situation is either true or false. The labeling step is done by humans and is therefore extremely time-consuming. Moreover, labeling of higher-level situations is not always straightforward, as critical situations can have very similar patterns than non-critical situations.

- *Data Selection:* It has to be decided, which of the labeled data should be used for training the model. There has to be enough data to represent several variations of the situation. Especially when the model has many parameters, there has to be enough training data in order to avoid overfitting.
- *Model Validation:* It has to be guaranteed that the trained model is able to recognize situations under various circumstances. So there should be a kind of testbed for evaluating the model under real circumstances to determine the reliability and trustworthiness of the results when using the trained model.

Thus, the process of generating training data for one situation of interest is quite complex and time-consuming. Moreover, this process has to be done for every situation of interest. But especially in security-related applications, situations of interest can change very rapidly in their characteristics and they do not happen very often. This means that in most cases, there is only few or even no training data available. Therefore, the machine learning approach is not fully satisfactory for detecting such situations, even if these methods are very powerful.

For human experts instead, it is quite easy to formulate and define the characteristics for new situations of interest. Therefore, our approach is to model this expert knowledge instead of using machine learning approaches. For modeling the expert knowledge and recognizing the situations of interest, we use a probabilistic model, i.e., a Dynamic Bayesian Network (DBN). DBNs, and especially their simplified version, the Hidden Markov Models (HMM) are widely used in machine learning approaches for situation recognition. The potential of these models is for example shown in [6] for maritime surveillance and in [7] for traffic scenarios. It has been shown that these models are able to handle noisy input data like wrong observation values.

However, maritime experts are in general not familiar with such kind of models. They are not able to determine the parameters of the model, i.e., the conditional probabilities, which are the crucial point for the model to yield the expected results. In order to support maritime experts in modeling their own situations of interest, we present an approach for setting the parameters automatically, based on the structure of the DBN. The structure of the DBN is generated based on a Situational Dependency Network (SDN), a human-understandable modeling of the situations of interest. The most challenging task is to determine the parameters in a way that the resulting DBN behaves like the human expert would expect it to behave.

The paper is structured as follows. In Section II, an overview of related work is given. The information flow in intelligent surveillance systems is highlighted in Section III. A detailed description of the situation assessment process is given in Section IV. Section V covers a definition of the term situation and it is shown how situations of interest can be characterized and how they can be represented in a SDN. Section VI starts with a brief review of DBNs and it is shown how the existences of situations of interest can be inferred. In Section VII, the approach of generating the DBN, especially the parameters, is presented. Finally, an application scenario in the maritime domain is given in Section VIII and it is shown that the approach yields the expected results.

II. Related Work

Working with heterogeneous sensors, the theories of multi-sensor data fusion [8] offer a powerful technique for supporting the situation assessment process. A lot of research has been done in combining object observations coming from different sensors [9], and also in the development of real-time methods for tracking moving objects [10]. Regarding data fusion in surveillance systems, the *object-oriented world model (OOWM)* is an approach to represent relevant information extracted from sensor signals, fused into a single comprehensive, dynamic model of the monitored area. It was developed in [11] and is a data fusion architecture based on the JDL (Joint Directors of Laboratories) data fusion process model [12]. Detailed description of the architecture and an example of an indoor surveillance application has been published in [13]. The OOWM has also been applied for wide area maritime surveillance [14].

In [15], an overview of different approaches of modeling and recognizing situations in the area of pervasive computing is presented. However, these approaches can also be used in the area of surveillance systems. In his article, Ye et al. divide between two main techniques, specification-based techniques and learning-based techniques. DBNs are not addressed directly in his article, only HMMs, which are a special case of DBNs. However, HMMs are usually used for learning-based approaches. We will not list machine learning approaches here, as they are out of scope of this paper, but we refer to [15] for an extensive discussion on them.

Specification-based techniques are used, when humans model the situations of interest directly. We refer to them as expert modeling approaches. A very recent work using specification-based techniques has been published in [16], in which petri nets are used for modeling and recognizing situations of interest. In [17], situation modeling is done by situational graph trees and situation recognition is performed based on fuzzy-metric temporal logic. However, most of the specification-based techniques are based on ontological modeling, see for example [18], [19]. The reasoning in ontologies is performed by queries based on description logic and there exist several publications on the detection of maritime situations with ontologies, e.g., [20], [21], [22], or [23]. The main drawback in ontological modeling is that it is not possible to deal with uncertain information. For this reason, probabilistic reasoning mechanisms, as provided by Bayesian networks, are often used for situation assessment, e.g., [24] and [25].

Another development is the extension of the Web Ontology Language (OWL), see [26], with a probabilistic representation, which results in the Probabilistic Web Ontology Language

(PR-OWL) presented in [27]. PR-OWL applies the theory of Multi-Entity Bayesian Networks (MEBN) that have been developed by Laskey [28]. MEBN are an extension of Bayesian networks in the form that they allow representation of graphical models with repeated sub-structures. Applications of PR-OWL and MEBNs are presented in [29] for the semantic web, and in [30], [31] for maritime applications. In [28], an algorithm was presented, which creates situation-specific Bayesian networks, based on the modeled MEBN. However, the main drawbacks in the MEBN-approach are that the parameters of the network still have to be inserted manually and that it is not possible to generate DBNs. However, the situation assessment should be able to deal with noisy observations and thus, the method of DBNs should be used. The work of this artice is inspired by the MEBN-approach and addresses the aforementioned problems.

First ideas of modeling situations in surveillance applications have been presented in our previous work in [32]. In [33], a Bayesian network, was applied to observed objects in the maritime domain and the user acceptance of such an automatic situation assessment was shown. Further work has been done in including temporal dependencies into the model, i.e., by defining a DBN for the detection of vessels that are most likely to carry refugees on board, and it was shown that the DBN-approach yields promising results, see [1]. As modeling a DBN is a difficult task and maritime experts are in general not familiar with probabilistic methods, as they are not able to determine the parameters of the model, a method for an automated generation of a DBN was developed and presented in [34] for scenarios on a parking space. A similar approach was then applied to the maritime domain in [35]. This paper extends the previous developed concepts for characterizing situations at different abstraction levels and the methods for generating a DBN.

III. Information Flow in Surveillance Systems

In this section, we will describe the information flow in surveillance systems in a general way. The general information flow for intelligent surveillance systems is visualized in Figure 1, wherein information aggregates are represented by boxes, and processes are represented by circles. The information flow is as follows.

In surveillance applications, the task is to observe a spatio-temporal section of the real world, a so-called *world of interest*. We will term all elements in the world of interest *entities*. By the term entity, not only physical objects like vessels are meant, as entities can also be non-physical elements in the real world like vessel attributes, relations between vessels or situations. Furthermore, not all entities can be observed, as there is no sensor to observe them directly. Thus, entities can represent observable or non-observable elements. All entities together represent the complete state of the world of interest.

The next step in the information flow is the *observation* of entities by several sensors. Sensor systems for observing the real world can be of extremely heterogeneous types, e.g., video cameras, infrared cameras, radar equipment, or radio-frequency identification (RFID) chips. Even human beings can act like a sensor by observing entities of the real world. Observing the world of interest with sensors results in *sensor data*, for example a radar image or a video stream.

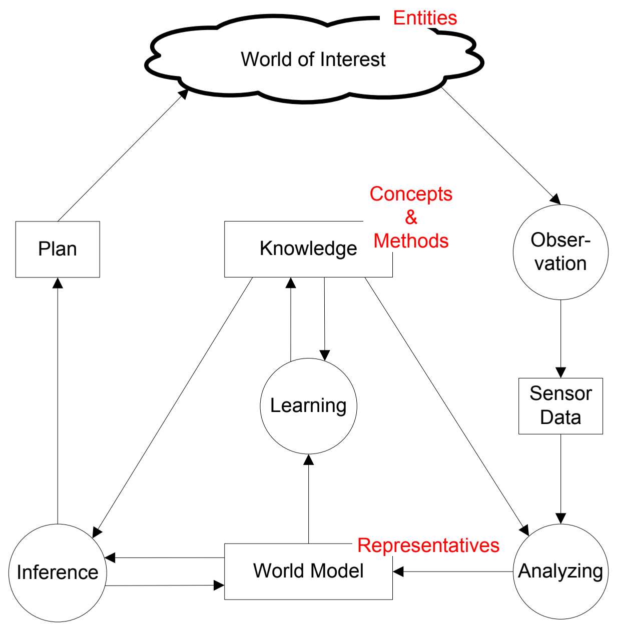


Fig. 1. Information flow in a surveillance system represented by information aggregates *(boxes)* and processes *(circles)*.

Analyzing sensor data is done by means of knowledge and the resulting information is transferred to the world model. Analyzing sensor data includes for example the detection and localization of moving vessels at sea from a video stream. Knowledge contains all information that is necessary for analyzing sensor data, for example specific signal-processing methods and algorithms used for the detection, localization and tracking of vessels in video streams.

The *world model* is a representation of entities in the world of interest and consists therefore of *representatives*. Every representative has a corresponding entity in the real world. The mapping between entities in the world of interest and representatives in the world model is structure-preserving and can therefore be interpreted as a homomorphism. Specific mappings are defined by concepts, which are part of the knowledge. Concepts are for example used in the analyzing process by defining how an observed vessel is represented in the world model. As the world of interest is highly dynamic and changes over time, the history of the representatives is also stored in the world model.

However, as mentioned before, some entities cannot be observed directly. Therefore an *inference* process is reasoning about non-observable (and also unobserved) entities by means of knowledge. A simple inference process is for example to calculate an object's velocity from the last and current position. A more complex inference process would be to estimate if the intention of an observed vessel is benign or adversarial. Doing this way, the world model is always being updated and supplemented with new information by predefined inference processes. Thus, during operation, the world model tries to estimate a complete representation of the world of interest in every time step.

Summing up, *knowledge* contains all information for analyzing sensor data, updating the world model and supplementing it with new information. Knowledge consists of abstract *concepts* and also of *methods*. Concepts are used for the representation of real-world entities in the world model and methods are used for analyzing data or inferring further information. Characteristics of the knowledge are of course extremely dependent on the application domain. Additionally, knowledge is not static. The content of the world model can be used for acquiring new knowledge by a *learning* process. This could be, for example, a method for structure or parameter learning in probabilistic graphical models.

To close the loop of the information flow, the result of an inference process could also include a *plan* of how to act further in the real world. Thus, the inference process can also act like a decision support, on which the action plan is based. The plan itself can be an action plan for an agent, for example, to call the police, or a sensor management plan, for example, a request for more detailed information from a special sensor.

Finally, we have to mention that the presented information flow in surveillance systems is not intended to act fully automatically. Every process can be designed in a way that a human operator is involved and that he is able to use the system interactively.

In the next section, we will have a more detailed look at a specific inference process, namely the situation assessment process. The situation assessment process tries to estimate predefined situations of interest by using the information of assessed objects over time.

IV. The Situation Assessment Process

By situation assessment, we mean the process of estimating the existence of situations of interest, which is conform to [2] and [12]. We divided the whole process into several sub-processes, as we state that situation assessment does not only consist of the process of recognizing situations, but also of the process of characterizing and modeling situations of interest, based on the current task. The conceptual framework of the process is depicted in Figure 2 and will be described in the following.

During the process of *object assessment*, estimates of objects are created, which do not only include kinematic state estimates like tracking the position and velocity of vessels, but also descriptive attributes of the object. The result of this process are *object representatives*, which are stored in the world model over time. Thus, we will first describe how the concept of an object is defined, i.e., how it is stored in the world model.

The concept of an object is defined as a physical entity of the real world. An object belongs to exactly one object class. As an object has several attributes, an object class is defined as the equivalence class of identical attribute lists. The attributes of the object can be divided into properties and states. Properties are time-invariant attributes, e.g., the length or the name of a vessel. State values can change over time and are therefore time-variant, e.g., the position or the velocity of a vessel. Regarding its spatial position, an object can be mobile, e.g., a vessel, or stationary, e.g., a land border. Thus, for a vessel the position is a state attribute and for the land border the position is a property. The concept of an object is visualized in Figure 3.

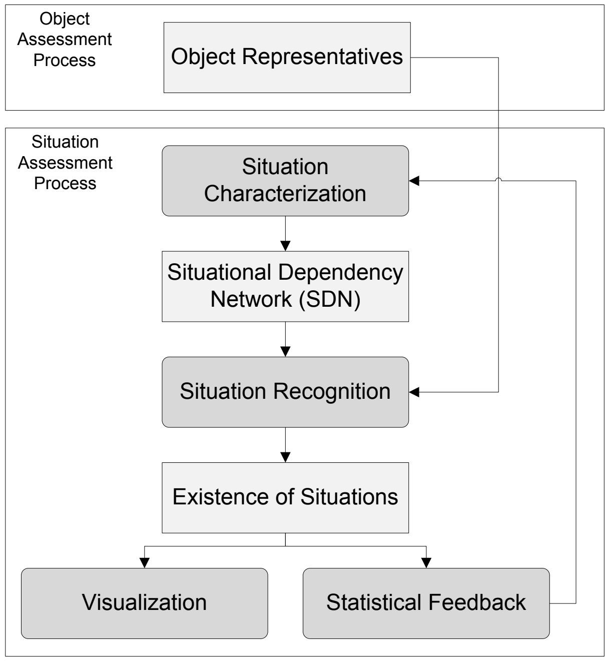

Object Assessment Process
Object Representatives
Situation Assessment Process
Situation Characterization
Situational Dependency Network (SDN)
Situation Recognition
Existence of Situations
Visualization
Statistical Feedback

Fig. 2. The process of situation assessment.

As the representation in the world model also has a memory, which means that the past states of an object are stored, the complete history of the observed object is always available. As this operation is very memory-intensive, for practical reasons the world model should be connected to a database and only the latest objects should be hold in memory. Furthermore, the representation of an object in the world model does not only include observed attributes, but also inferred ones. For example, based on observed positions of a vessel, the velocity can be inferred. Furthermore, attribute values can be quantitative or qualitative. For example, the absolute position and velocity of a vessel are quantitative attributes, and the attribute value that a vessel is made of wood is a qualitative one. The selection of the attributes is still up to the user. However, the selected attributes should be dependent on the sensor capabilities and also on the required task of the situation assessment.

The process of situation assessment is divided into the sub-processes of situation characterization, situation recognition, visualization, and statistical feedback, which will be explained in the following. The first sub-process of *situation characterization* includes the a-priori modeling of expert knowledge about situations of interest. Because there are a lot of semantic dependencies between situations, these have to be modeled in order to estimate the existence of the situations correctly. This sub-process results in a *Situational Dependency Network (SDN)*. We will address the process of characterizing situations in detail in Section V.

The sub-process of *situation recognition* analyses the object representatives over time with respect to the existence of the

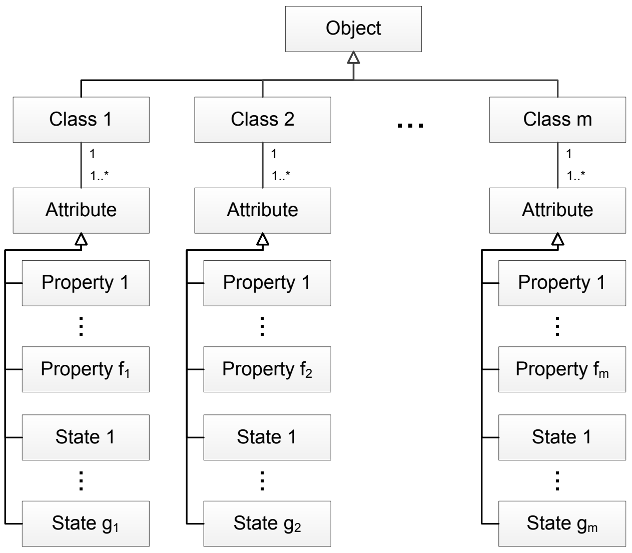


Fig. 3. The concept of an object.

situations of interest. Thus, it applies the SDN to the estimated object properties and states and infers, if the situations of interest are existing or not. The result of this process is a set of existence probabilities, one existence probability for each situation of interest. As we use DBNs for the process of situation recognition, the existence probabilities can be calculated by state-of-the-art inference mechanisms and algorithms that have been developed for DBN. Inference can be performed either exact or approximate. We refer the reader to [36] for detailed information on such algorithms, as this is out of scope of the article. However, many algorithms are available and ready to use, e.g., in specific software [37] or Matlab toolboxes [38]. Based on the resulting probabilities, the existence of situations can be inferred, if the probability value exceeds a certain threshold value.

This calculation, combined with an appropriate *visualization*, allows for a prompt assessment of the whole situation and thus for prompt decisions. Inferred situations of interest can be visualized in a dynamic map, where the involved vessels are highlighted. Also the calculated existence probability of the situations of interest can be visualized additionally. Of course, visualization is a crucial point of supporting situation awareness. Even if the results are calculated correctly, the visualization of them to the user during operation can be done in a wrong way, e.g., by presenting him too much information. This results in an information overload and thus, reduces his situation awareness [2]. However, evaluating different visualization methods for situations of interest is out of scope for this article and is therefore not discussed in detail.

Furthermore, a *statistical feedback* can be calculated from the set of existence probabilities over time. This can be used for refining the process of situation characterization and thus is a process for integrating the user in the situation assessment process. The feedback can have several objectives, for example, it could suggest a refinement of the SDN or could indicate situations that have not been detected at all, i.e., ask the user if the situation is still of interest.

V. Characterizing Situations of Interest

In this section, we will describe how situations can be modeled as random variables and how their existence is defined. We will introduce two different situation types and how the dependencies between several situations can be modeled. Finally, we will show how a complete SDN can be modeled.

A. Situation Modeling

Before we are able to characterize situations of interest, we have to define the term situation. In [15], a situation is defined as follows:

> “A situation is defined as an external semantic interpretation of sensor data. Interpretation means that situations assign meanings to sensor data. External means that the interpretation is from the perspective of applications, rather than from sensors. Semantic means that the interpretation assigns meaning on sensor data based on structures and relationships within the same type of sensor data and between different types of sensor data.”

This definition corresponds to our understanding of the term situation, but we try to give a more formal definition of a situation. First of all, we state that a situation at time point t is always connected with an external semantic statement, which is either true or false. The semantic statement is always based on a temporal sequence of a specific constellation of modeled objects and their attributes.

We define $O^1, O^2, \dots, O^n$ as the set of objects that are relevant for the semantic statement. We define $A_1^i, A_2^i, \dots, A_{m_i}^i$ as the set of the relevant attributes of the object O^i, with $i = 1, \dots, n$. If an object or an attribute is relevant or not is induced by the semantic statement of the situation and has to be defined by the user. For example, for the situation that a vessel is fast, only it’s velocity is relevant and it’s heading can be ignored. We can then define the configuration space $\mathcal{O}$ as

$$\mathcal{O} = \bigtimes_{i=1}^{n} \bigtimes_{k=1}^{m_i} r(A_k), \tag{1}$$

where $r(A_k)$ denotes the range of the attribute values of A_k. $\mathcal{O}$ has then the dimension $\dim \mathcal{O} = \sum_{i=1}^{n} m_i$.

We set $\Omega = \mathcal{O} \times T$, where T represents the time and define a situation S_t at time t as the mapping

$$S_t : \Omega \rightarrow \{0, 1\}. \tag{2}$$

We say that the set $\tilde{\Omega} = \widetilde{\mathcal{O}} \times \widetilde{T} \subseteq \mathcal{O} \times T$ is the support of the Situation S_t, if

$$S_t(\omega) = \begin{cases} 1, & \text{if } \omega \in \widetilde{\Omega}, \\ 0, & \text{if } \omega \notin \widetilde{\Omega}. \end{cases} \tag{3}$$

Example 1: The semantic statement is that two objects are close to each other, e.g., a yacht is close to a tanker. Relevant objects are A and B, with a one-dimensional position value

x_a and x_b, respectively. We chose a one-dimensional position for visualization purposes. It is $\Omega = \mathcal{O} \times t$, where t is a time point. Then it is

$$\omega \in \widetilde{\Omega} \Leftrightarrow |x_a - x_b| \leq r, \tag{4}$$

where r is the threshold value. The support of this situation is visualized in Figure 4.

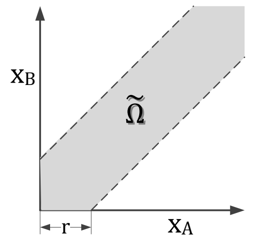


Fig. 4. The support of the situation that object A is close to object B.

Example 2: The semantic statement is that two objects are approaching each other, e.g., a vessel is approaching a specific harbor. Relevant objects are A and B, with a one-dimensional position value x_a and x_b, respectively, where x_b is static. It is $\Omega = \mathcal{O} \times T$, where T is a discrete time interval. Then it is

$$\omega \in \widetilde{\Omega} \Leftrightarrow |x_a - x_b|_t < |x_a - x_b|_{t-1}, \forall t, t-1 \in T. \tag{5}$$

The support of this situation is visualized in Figure 5.

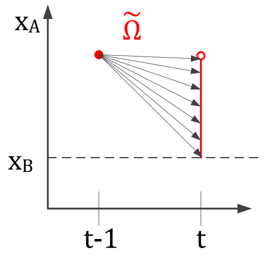


Fig. 5. The support of the situation that object A is approaching object B, with a selected value x_A for time point $t-1$.

B. Existence of Situations

We say that a situation exists, if the semantic statement is true, and that it does not exist, if the semantic statement is false. Thus, a situation exists, if the observed objects can be assigned to the relevant objects of the configuration space and for their attribute values it holds $\omega \in \widetilde{\Omega}$.

For modeling the existence of situations, we interpret the situation, i.e., the mapping $S_t : \Omega \to \{0,1\}$ as a binary random variable. It is P a probability measure on $S_t(\Omega)$, thus a probability distribution of S_t. Then the existence of a situation S_t at time point t is given by $P(S_t = 1)$, or shortly $P(S_t)$. Thus, when performing situation recognition, we are interested in the probability $P(S_t)$.

Due to this modeling, situations are characterized by information collected over a time-period, but they only exist at a special point in time. Their existence in the next time-point has to be verified again. The time-period itself is induced by the semantic statement of the situation. In example 2, it is enough to take $[t-1, t]$ into account. But for other situations it may be necessary to expand the time-period, e.g., for the situation that the vessel was in a suspicious area during the last 2 hours.

C. Different Situation Types

We now face the problem that the semantic statement of situations can be arbitrary complex and on a high abstraction level, e.g., a vessel is a suspicious smuggling vessel. Thus, the recognition of the situation depends on various characteristics and the direct modeling of the support of the situation is not always possible. Because of this fact, we can differentiate between the following two cases:

- The existence of a situation can imply the existence of other situations.
- The existence of a situation can lead to the existence of another situation.

These dependencies can be used for the recognition of situations. To use them, we will define two types of situations, namely elementary situations and abstract situations:

- *Elementary situations*: The support of the situation can be modeled directly. The existence of the elementary situation S_t can be modeled as the deterministic mapping
$$P(S_t|\omega) = \begin{cases} 1, & \text{if } S_t(\omega) = 1, \\ 0, & \text{if } S_t(\omega) = 0. \end{cases} \tag{6}$$
- *Abstract situations*: It is not possible to model the support of the situation directly. The existence of the abstract situation S_t is dependent of the existence of n other (elementary or abstract) situations $S_t^1, S_t^2, \ldots, S_t^n$
$$P(S_t|S_t^1, S_t^2, \ldots, S_t^n) = \frac{P(S_t, S_t^1, S_t^2, \ldots, S_t^n)}{P(S_t^1, S_t^2, \ldots, S_t^n)}. \tag{7}$$

However, with an increasing number of dependent situations, it is not possible for an expert to model the joint probability distribution $P(S_t, S_t^1, S_t^2, \ldots, S_t^n)$. The solution for this is to use the chain rule of probability by using the aforementioned dependencies, e.g., for two situations S^1, S^2 this would be:

$$P(S^1, S^2) = P(S^1|S^2)P(S^2) = P(S^2|S^1)P(S^1). \tag{8}$$

Thus, it is sufficient to model the conditional probabilities between dependent situations. An attempt to visualize these dependencies is depicted in Figure 6. Of course, in real applications, it is often not straightforward to identify the dependencies between situations. In practice, the identification of dependencies have to be done by experts together with system developers.

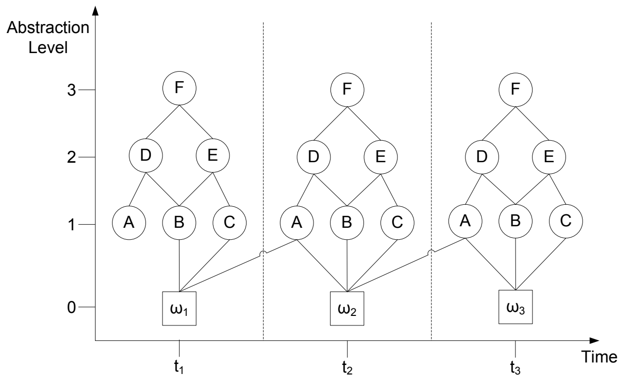


Fig. 6. Visualization of dependencies and abstraction levels, visualized without direction of dependencies.

D. Situational Dependencies

After defining situations formally, we have to consider their semantic interpretation, especially their relationships among each other. In [15], Ye et al. distinguish between five different types of relationships: generalization, composition, dependence, contradiction and temporal sequence, which we will repeat shortly in the following:

- *Generalization:* A situation is more general than another situation, if the occurrence of the latter implies that of the former.
- *Dependence:* A situation depends on another situation if the occurrence of the former situation is determined by the occurrence of the latter situation.
- *Composition:* A situation can be decomposed into a set of smaller situations, which is a typical composition relation between situations.
- *Contradiction:* Two situations can be regarded as mutually exclusive from each other if they cannot co-occur at the same time in the same place on the same subject.
- *Temporal sequence:* A situation may occur before, or after another situation, or interleave with another situation.

Our approach is now to model these relationships in a SDN. For the SDN, we divide the relationships into two main categories: sufficient and necessary conditions. We will explain them in the following:

- *Necessary condition:* A situation A is necessary for another situation B, if the existence of B implies the existence of A, i.e.,
$$B \xrightarrow{\text{N}} A.$$
If we have more than one necessary situations $A_1, \dots, A_n$, we have
$$B \xrightarrow{\text{N}} A_1 \wedge A_2 \wedge \dots \wedge A_n.$$
- *Sufficient condition:* A situation A is sufficient for another situation B, if the existence of A implies the existence of B, i.e.,
$$A \xrightarrow{\text{S}} B.$$
If we have more than one sufficient situations $A_1, \dots, A_n$, we have
$$A_1 \vee A_2 \vee \dots \vee A_n \xrightarrow{\text{S}} B.$$

Thus, we can always interpret the arrow from A to B as follows: If situation A exists, then situation B exists. Or in logical notation: $A \Rightarrow B$, namely A implies B. $A \xrightarrow{\text{N}}$

Compared to the five different types of relationships defined in [15], we can state the following:

- *Generalization:* The generalization is in our case modeled as the sufficient condition.
- *Dependence:* The dependency is in our case modeled as the necessary condition.
- *Composition:* The composition is in our case modeled through the necessary condition with more than one necessary situation.
- *Contradiction:* The contradiction is not explicitly modeled in our case, but should, of course, be represented by the semantics of the model. This means, the parameters of the DBN should be determined in a way that the two situations cannot exist simultaneously.
- *Temporal sequence:* The temporal sequence is not yet addressed in our model so far. However, we can extend the model by defining a specific sequence of existences of different situations as a situation of interest. We can recognize this sequential situation, for example, by comparing it to the results of the Viterbi-algorithm, which calculates the probability of the most likely sequential situation.

E. Situational Dependency Network

We further assume that in real world applications, sensor observations will be noisy. Noisy observation data can appear as wrong observations or as missing observations. It is also possible that some modeled elementary situations cannot be observed because there is no sensor available that would be able to observe the necessary information. Of course, the fact of noisy observations has a big influence on the situation recognition process.

Thus, we want to achieve a kind of inertia in the process of situation recognition. The inertia of the process supports the following statements:

- If we observe an area of interest over time, a single observation should not yield to the recognition of a situation, as it could be a noisy observation.
- If a situation of interest exists in the time step before, it is very likely that it exists in the current time step, also if different observations are made.

For achieving this inertia in the process, we have to extend our model with recursive, i.e., temporal arrows. We will add the temporal arrow for abstract situation of interest, whose elementary situations are assumed to be noisy, i.e., visually draw an arrow from the situation at time point t to the same situation at time point $t + 1$. The temporal arrows can then be used in the DBN and result in a filtering effect of the existence probability. As we will show later, the strenght of the filtering effect can be adjusted by different weights. These weights will have an influence on how many observations have to be made to justify the existence of a situation or how many noisy observations can be made without rejecting the existence.

Example 3: In this example, we have three situations:

- Situation *area*: The vessel was in a suspicious area, known for smuggling activities.
- Situation *AIS*: The vessel is not sending any self-identification signal like AIS.
- Situation *smuggling*: The vessel is a suspicious smuggling vessel.

The first and the second situations (area and AIS) are elementary situations and the third one (smuggling) is an abstract situation, which we are interested in, and it is dependent on the two others. The dependencies are as follows. If a vessel was in a suspicious smuggling area, it is very likely that the vessel is a smuggling vessel. Thus, the area situation is a sufficient condition for the smuggling situation and we draw an arrow from area to smuggling. If a vessel is a smuggling vessel, then it is pretty sure that it does not send any identification signal. Thus, the AIS situation is a necessary condition for the smuggling situation and we draw an arrow from smuggling to AIS. As we assume noisy observations, we add a temporal arrow to the smuggling situation. The overall SDN is depicted in Figure 7, where the temporal arrow is indicated with a red T in the lower right corner of the node. Note that the same SDN can be used, even if the suspicious area itself can change.

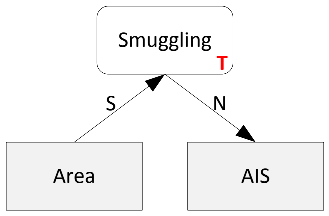


Fig. 7. Example of a situational dependency network.

Finally, we can present the general approach for modeling the SDN in Algorithm 1 and the calculation of the abstraction level in Algorithm 2.

VI. Recognizing Situations of Interest

Due to this modeling, the SDN can be interpreted as a probabilistic graphical model, namely a DBN. In a simple Bayesian network, the basic idea is to decompose the joint probability of various random variables into a factorized form. We will now describe how a DBN is defined and how the existence probabilities of the modeled situations can be inferred.

Algorithm 1 Creating a SDN

```
Require: set of situations and known pairwise dependencies
Ensure: SDN
  model situations as nodes
  for all nodes do
    if support can be modeled directly then
      declare it as elementary situation
    else
      declare it as abstract situation
    end if
  end for
  for all dependencies do
    if situation $A$ is sufficient for situation $B$ then
      model the dependency as an arrow from $A$ to $B$
    end if
    if situation $A$ is necessary for situation $B$ then
      model the dependency as an arrow from $B$ to $A$
    end if
  end for
  for all abstract situations do
    if abstract situation is dependent on an elementary situation with noisy observations then
      add a temporal arrow to the abstract situation
    end if
  end for
```

Algorithm 2 Calculating abstraction levels

```
Require: SDN
Ensure: level of abstraction for each node
  for all elementary nodes do
    abstraction level =1
  end for
  i=2
  initialize $\widetilde{\boldsymbol{S}}$ with a non-empty set
  while $\widetilde{\boldsymbol{S}} \neq \emptyset$ do
    set $\widetilde{\boldsymbol{S}}$ to the set of abstract situations that are only dependent on situations with abstraction level $i-1$
    set all elements of $\widetilde{\boldsymbol{S}}$ to abstraction level $i$
    set $i = i + 1$
  end while
```

A. Dynamic Bayesian Networks

In a Bayesian network, random variables $X^1, X^2, \dots, X^n$ are depicted as nodes and conditional probabilities as directed edges. The joint probability can then be factorized as

$$P(X^1, \dots, X^n) = \prod_{i=1}^{n} P(X^i | Pa(X^i)), \tag{9}$$

where $Pa(X^i)$ is the set of parents of the node X^i. If $Pa(X^i)$ is an empty set, then X^i is a root node and $P(X^i|Pa(X^i)) = P(X^i)$ denotes its prior probability.

A DBN [39] is defined as a pair $(B_0, 2TBN)$, where

- B_0 defines the prior distribution $P(\boldsymbol{X}_0)$ over the set $\boldsymbol{X}_0$ of random variables, and
- $2TBN$ defines a Bayesian network over two time

slices with

$$P(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) = \prod_{i=1}^{n} P(X_t^i|Pa(X_t^i)), \qquad (10)$$

where X_t^i is a node at time slice t and $Pa(X_t^i)$ is the set of parent nodes, which can be in the time slice t or in the time slice $t-1$.

Note that in the definition of a $2TBN$, $Pa(X_t^i)$ is never empty, i.e., every node in time slice t has at least one parent node and, therefore, the left side of equation (9) differs from the left side of equation (10). An example of a $2TBN$ with 3 nodes in each time slice is shown in Figure 8.

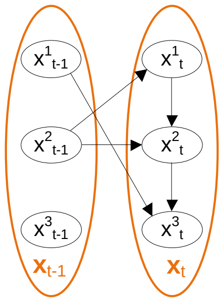


Fig. 8. An example of a $2TBN$ defining dependencies between two time slices and dependencies between nodes in time slice t adopted from [10]. Note that a $2TBN$ does not define the dependencies between nodes in time slice $t-1$.

The joint probability distribution of a DBN can then be formulated as

$$P(\boldsymbol{X}_{0:T}) = P(\boldsymbol{X}_0) \cdot \prod_{t=1}^{T} \prod_{i=1}^{n} P(X_t^i|Pa(X_t^i)), \qquad (11)$$

with $P(\boldsymbol{X}_{0:T}) = P(\boldsymbol{X}_0, \ldots, \boldsymbol{X}_T)$.

As we want to model a network of situations by a DBN, the structure of the network has to fulfill the following assumptions:

- Stationarity: the dependencies within a time slice t and the dependencies between the time slices $t-1$ and t do not depend on t.
- 1st order Markov assumption: the parents of a node are in the same time slice or in the previous time slice.
- Temporal evolution: dependencies between two time slices are only allowed forward in time, i.e., from past to future.
- Time slice structure: The structure of one time slice is a simple Bayesian network, i.e., without cycles.

If any of these assumptions are not fulfilled, the network is not a DBN and inference algorithms could not be applied.

B. Inferring Existence Probabilities

Due to the dependency between elementary and abstract situations and the fact that we can feed the DBN with evidence, i.e., observations, only via the elementary situations, we can calculate the joint probability recursively in time by

$$P(\boldsymbol{S}^*_{0:T}, \boldsymbol{E}_{1:T}) = P(\boldsymbol{S}^*_0) \cdot \prod_{t=1}^{T} P(\boldsymbol{S}^*_t|\boldsymbol{S}^*_{t-1}) P(\boldsymbol{E}_t|\boldsymbol{S}^*_t), \quad (12)$$

where $\boldsymbol{E}$ denotes the set of elementary situations filled with evidences, and $\boldsymbol{S}^*$ denotes all defined situations $\boldsymbol{S}$ in the DBN without the collected evidence nodes, i.e., $\boldsymbol{S}^* = \boldsymbol{S} \backslash \boldsymbol{E}$.

By using this kind of recursive calculation, we can make different calculations over time, which we list in the following. Note that $\widetilde{\boldsymbol{S}}$ is now an arbitrary set of abstract situations.

- Filtering: $P(\widetilde{\boldsymbol{S}}_t|\boldsymbol{E}_{1:t})$ gives a solution to the existence probability of a set of situations $\widetilde{\boldsymbol{S}}$ at the current time,
- Prediction: $P(\widetilde{\boldsymbol{S}}_{t+k}|\boldsymbol{E}_{1:t})$ (with $k > 0$) gives a solution to the existence probability of a set of situations $\widetilde{\boldsymbol{S}}$ in the (near) future,
- Smoothing: $P(\widetilde{\boldsymbol{S}}_k|\boldsymbol{E}_{1:t})$ (with $0 < k < t$) gives a solution to the existence probability of a set of situations $\widetilde{\boldsymbol{S}}$ in the past,
- Most likely explanation: $\text{argmax}_{\widetilde{\boldsymbol{S}}_{1:t}} P(\widetilde{\boldsymbol{S}}_{1:t}|\boldsymbol{E}_{1:t})$ gives a solution to the most likely sequence of situations $\widetilde{\boldsymbol{S}}_{1:t}$.

Due to this modeling, the existence probability of an arbitrary set of abstract situations can be calculated in a recursive way at each point in time. A situation is then represented in the world model, if the corresponding existence probability is larger than an instantiation-threshold. If the existence probability in the next time step is below a deletion-threshold, it is assumed that the situation does not exist any longer and its representation is removed from the world model. This way, the world model tries to keep an up-to-date representation of the existing situations of the real world. However, determining a meaningful instantiation- and deletion-threshold is still an open task.

VII. Generating Situation-Specific DBNs

In this section, we will describe how we generate a DBN from our SDN model in order to be able to apply the aforementioned inference calculations. As we may have modeled a lot of situations in our SDN, and the operator may be only interested in a subset of them, we will generate a DBN with only the necessary nodes. We will term the generated DBN a *situation-specific DBN*, which we adopted from the MEBN-approach presented in [28].

A. SS-DBN Structure

For generating the structure of the SS-DBN, we will make use of the structure of the pre-modeled SDN. In general, we will use exactly the same structure, but only use the situations that are relevant for the selected situations of interest. The general approach is the described in Algorithm 3. Thus, we get a reduced version of our SDN by making use of the predefined abstraction levels. This way, we simply get our SS-DBN structure by selecting the nodes with a lower abstraction level.

Algorithm 3 Generating the SS-DBN structure

```
Require: SDN
Ensure: DBN-structure
  set S as selected situations of interest in the SDN
  while S $\neq \emptyset$ do
    for all incoming and outgoing arrows of every situation
    $s$ in S do
      if abstraction level of the connected node is lower than
      the one of $s$ then
        add node to DBN
      else
        ignore them
      end if
      set S as the set of all added nodes
    end for
  end while
  for all temporal nodes do
    add a reflexive arrow
  end for
```

B. Criteria for good Parameters

The main challenge is now to set the parameters in a way that the network behaves as we would expect it to behave. In the following, we will list some criteria, which the DBN should fulfill:

- *Asymptotic behavior:* How does the DBN behave if we observe the same values over time? Especially if we make no observations at all, we do not know anything about the existence of the situation and the resulting existence probability should be converging to 0.5.
- *Switching behavior:* How does the DBN behave if observations change their values significantly? If we first observe values that support the existence of the situation of interest and then we make the opposite observations, the resulting probabilities of the situation should also change, i.e., the situation should switch its state from true to false.
- *Robustness:* How does the DBN behave if we have wrong or missing observations? As said above, the resulting probability of the situation of interest should not be very sensitive for noise.

Having determined these criteria, we will evaluate several parameter settings with respect to them.

C. SS-DBN Parameters

We have to define parameters for every node, i.e., a priori probabilities for nodes with no incoming arrows, and conditional probabilities for nodes with incoming arrows. We will treat the temporal arrows the same as all other incoming arrows. The conditional probabilities then correspond to $P(X_t^i|Pa(X_t^i))$ in equation (10). As a first rule for setting the parameters, we have the following:

- If the situation has no incoming arrows, i.e., is a root node, the prior probabilities are all set to 0.5.
- If the situation is a temporal situation, we set the probabilities inside the first time point to 0.5.

In many cases, elementary situations have only one incoming arrow. This is due to the fact that at the lowest level, there are the observation values of the DBN and they can always be considered as necessary conditions. Thus, we have to differentiate between two different types: either the parent node is temporal or not. For non-temporal nodes, the higher level situation can be interpreted as a semantic annotation to the observed value and therefore, we can set the conditional probabilities in a deterministic way:

- If the elementary situation A has one incoming arrow from a non-temporal situation B, the probabilities are set deterministic like in Table I.

TABLE I. CONDITIONAL PROBABILITY TABLE (CPT) FOR ELEMENTARY SITUATION WITH NON-TEMPORAL PARENT.

$P(A\|B)$	B	$\neg B$
A	1.0	0.0
$\neg A$	0.0	1.0

If we have a temporal parent, we will apply a non-deterministic conditional probability table. We will evaluate some different parameter settings before choosing them. For this small evaluation, we will only consider two nodes A and B, where the parent B is a temporal node. We are interested in good values for the CPT $P(A|B)$. Thus, we fix the CPT parameters for $P(B_{t+1}|B_t)$. We set the CPT values in a mirrored way, i.e.,

$$P(A|B) = P(\neg A|\neg B) \text{ and } P(\neg A|B) = P(A|\neg B). \quad (13)$$

This is because we do not want to have different influence when observing true or false. Note that we only have to change one value for our evaluation, namely $p = P(A|B)$, as $P(\neg A|B) = 1 - P(A|B)$. For the temporal node, we fix the value $P(B_{t+1}|B_t) = 0.9$, see right side in Table II.

We evaluate five different values for p, and we show three different results, namely the probability $P(B|A)$ over ten time steps. Figure 9 shows the resulting probability for similar observations, switching observations, and for some false observations. For the first observation sequence (Figure 9a), we would like to result in a high probability, so we discard $p = 0.6$. For the second one (Figure 9b), we would like to have a probability, that switches, but not rapidly, so we discard $p = 0.99$. For the third sequence (Figure 9c), we would like to have a probability that is not too sensitive to false observations, so we finally choose $p = 0.7$. We therefore have the following rule:

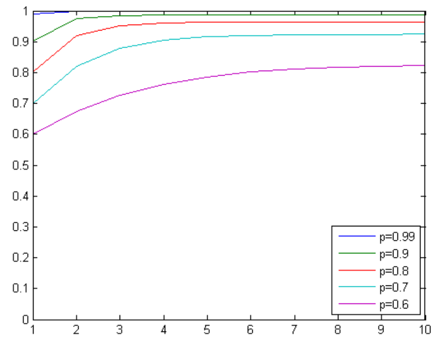
- If the elementary situation A has one incoming arrow from a temporal situation B, the probabilities are set like the ones on the left side in Table II.

TABLE II. CPT FOR ELEMENTARY SITUATION WITH TEMPORAL PARENT (LEFT SIDE) AND CPT FOR TEMPORAL PARENT (RIGHT SIDE).

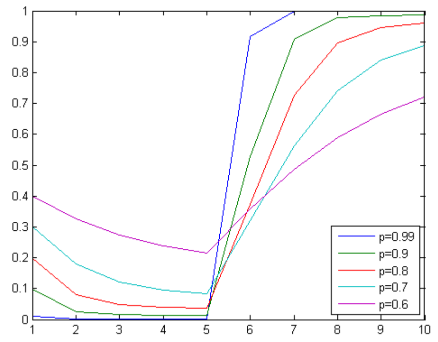
$P(A\|B)$	B	$\neg B$
A	0.7	0.3
$\neg A$	0.3	0.7

$P(B_{t+1}\|B_t)$	B_t	$\neg B_t$
B_{t+1}	0.9	0.1
$\neg B_{t+1}$	0.1	0.9

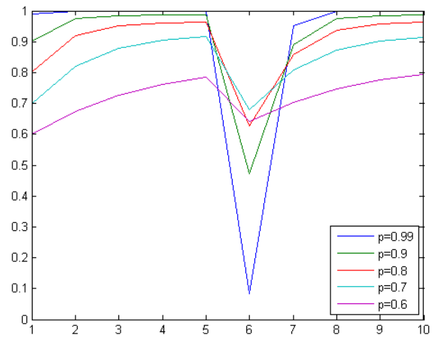
For all abstract situations, we apply an approach that uses a weighted CPT-construction. Let Y be the abstract situation, and $X_1, \ldots, X_k$ the situations with an arrow to Y.



(a) $A = 1111111111$

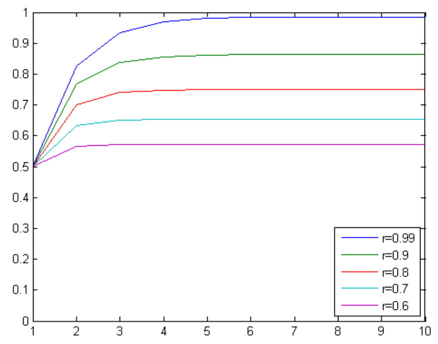


(b) $A = 0000011111$

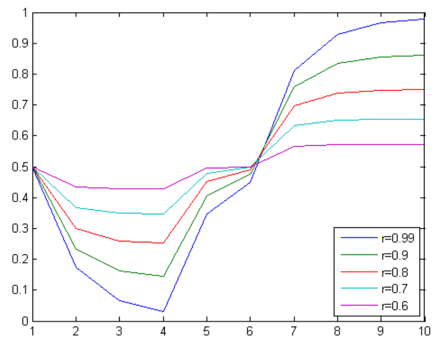


(c) $A = 1111101111$

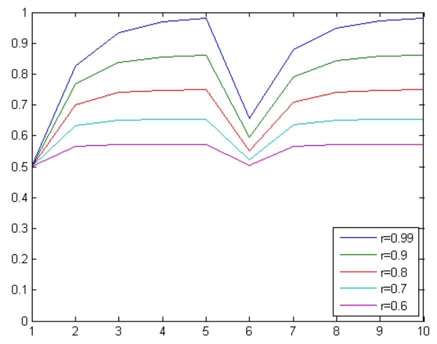
Fig. 9. $P(B|A)$ over ten time steps with different observation sequences.



(a) $A = 1111111111$, $B = 1111111111$



(b) $A = 0000111111$, $B = 0000001111$



(c) $A = 1111101111$, $B = 1111111111$

Fig. 10. $P(C|A,B)$ over ten time steps with different observation sequences.

For this approach, we have to determine two different types of parameters.

- Relative influence of variables: This is modeled by different weights λ_i with $\sum_{i=1}^{k} \lambda_i$.
- Absolute influence of variables: This is modeled by a stretch value $r \in [0.5, 1]$.

For the relative influence, we can use a weighting of the influence, namely

$$P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = \sum_{i=1}^{k} \lambda_i x_i. \tag{14}$$

Then it is $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) \in [0, 1]$ and $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = 0$ for $x_i = 0$ and $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = 1$ for $x_i = 1$.

The aim of the absolute influence is to reduce the interval $[0, 1]$ to the values of $[1 - r, r]$. Thus, the overall strength of the variables should be reduced. We define

$$f(x) = (2r - 1) \cdot x + (1 - r). \tag{15}$$

Then it is $f(x) \in [1 - r, r]$ for $x \in [0, 1]$. Thus, we can apply f on $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k)$ and we have $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) \in [1 - r, r]$. In summary, we have to determine $k + 1$ parameters, namely $\lambda_1, \ldots, \lambda_k, r$, instead of 2^{k-1} for the CPT of $P(Y = 1|X_1 = x_1, \ldots, X_k = x_k)$.

We will now show a small evaluation of our example with different values of r. The values of r are chosen with steps 0.1. As 0.5 and 1.0 would not be reasonable values, we chose the values 0.6, 0.7, 0.8, 0.9, and 0.99. We will use the same criteria as above, namely insert observation sequences that represent similar observations, switching observations, and for some false observations. The result is shown in Figure 10. For the first observation sequence (Figure 10a), we would like to result in a high probability, so we discard $r = 0.6$ and $r = 0.7$. For the second one (Figure 10b), we would like to have a probability, that switches, but not rapidly, so we discard $p = 0.99$. For the third sequence (Figure 10c), we would like to have a probability that is not too sensitive to false observations, so we finally choose $p = 0.9$. We can now state the final rule for setting the parameters:

- If the situation is on a higher level, the weighted CPT-construction is applied. The weights of the influences can be adapted due to the semantics, and it is suggested to set the stretch value to $r = 0.9$.

VIII. Application Example in the Maritime Domain

Assume for example a security officer who is using a maritime surveillance system located in a port on an island and he is interested in detecting vessels, which are suspicious smuggling vessels. There is also a suspicious zone next to the island, in which a lot of smuggling activities recently happened. The officer is able to formulate several characteristics that lead to a higher probability of a smuggling vessel, either if the vessel is incoming or outgoing. Note that these characteristics are defined as situations itself.

Based on the different situations, the expert is able to model an SDN, as depicted in Figure 11. A sketch of such situations is visualized in Figure 12. Let us give some examples for our modeling approach. In Figure 11, the situation `Has unknown ID` has the necessary characteristic situation `MMSI-Number is empty`. The situation `Sends no AIS signal` has the necessary characteristic situations `MMSI-Number is empty` and `Is inside AIS receiver area`. And the situation `Was in suspicious area` is a sufficient characteristic situation for the situation `Suspicious incoming smuggling vessel`. Thus, the existence of a sufficient situation should lead to a higher probability of the existence of the situation of interest, whereas the existence of the necessary situations has to be fulfilled for inferring the existence of the situation of interest.

Note that the arrows in the SDN are always pointing from a situation that describes some characteristics of the situation that is pointed to, either by a necessary or sufficient condition. The arrows of a necessary condition are always pointing from a higher level of abstraction to a lower level of abstraction and the arrows of a necessary condition vice versa. The level of abstraction of a single situation of interest is determined by the structure of the whole SDN. At the lowest level of abstraction, we only have our elementary situations that can be inferred as true or false directly from attribute values of objects or from geometric computations. An example is the situation

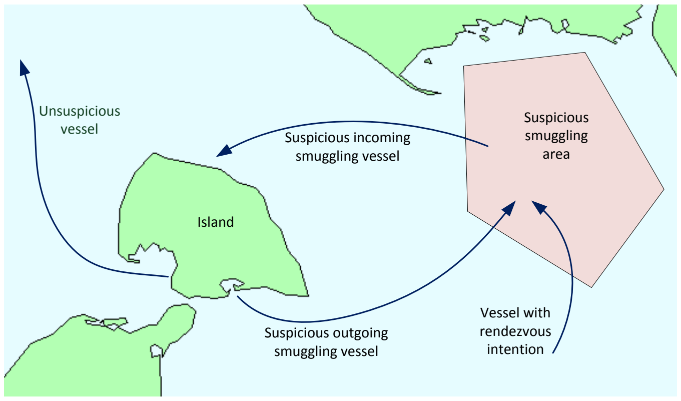

Unsuspicious vessel
Suspicious incoming smuggling vessel
Suspicious smuggling area
Island
Suspicious outgoing smuggling vessel
Vessel with rendezvous intention

Fig. 12. A sketch of the scenario for incoming and outgoing smuggling vessels.

`Past in polygon check` that checks, if any point of the vessel's path is inside the polygon of the suspicious area. Thus, at the lowest level, and only there, the observations are fed into network. Every situation with a direct connection, either incoming or outgoing arrows, is moved to the next higher level of abstraction. Based on these connections, we result in the SDN, as shown in Figure 11.

We will now evaluate the behavior of the whole network. We construct the DBN with the approach described above and set the parameters with the rules we established. We evaluate two different scenarios. In the first scenario the observations should lead to the decision of a suspicious outgoing smuggling vessel, the second one should indicate a suspicious incoming smuggling vessel.

For the first scenario, we will assume some observations that are not noisy, namely

- `past point in polygon check`=0 for all time points, i.e., the vessel was not in a suspicious area,

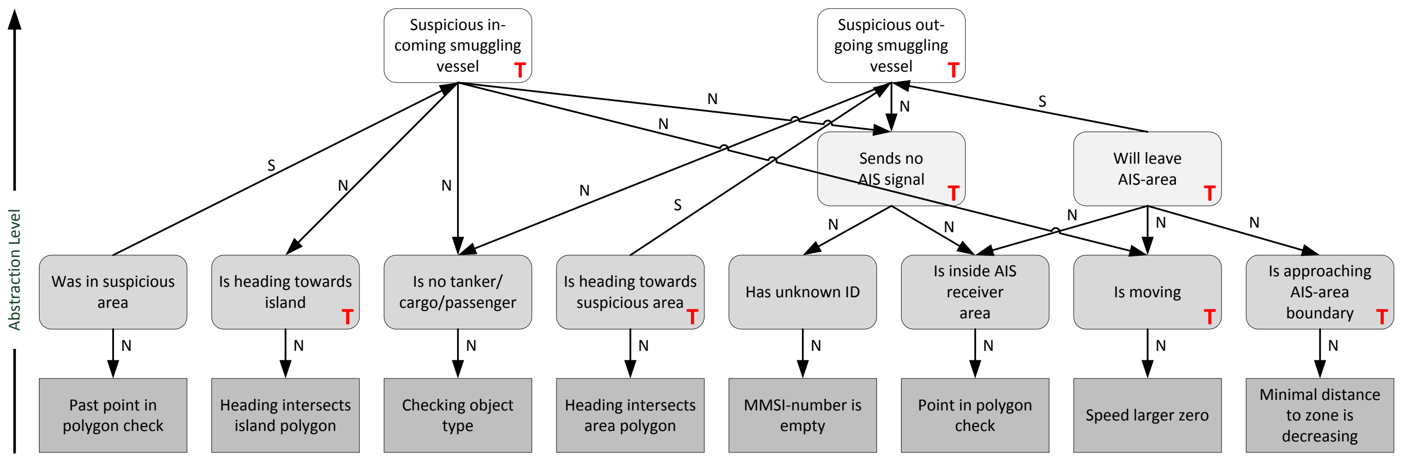

Abstraction Level
Suspicious in-coming smuggling vessel T
Suspicious out-going smuggling vessel T
Sends no AIS signal T
Will leave AIS-area T
Was in suspicious area
Is heading towards island T
Is no tanker/ cargo/passenger
Is heading towards suspicious area T
Has unknown ID
Is inside AIS receiver area
Is moving T
Is approaching AIS-area boundary T
Past point in polygon check
Heading intersects island polygon
Checking object type
Heading intersects area polygon
MMSI-number is empty
Point in polygon check
Speed larger zero
Minimal distance to zone is decreasing
S
N

Fig. 11. A SDN with two modeled situations of interest: suspicious incoming and suspicious outgoing smuggling vessel.

- `checking object type`=1 for all time points, i.e., the vessel is no tanker/cargo/passenger vessel,
- `MMSI-Number is empty`=1 for all time points, i.e., vessel has an unknown ID,
- `point in polygon check`=1 for all time points, i.e., vessel is inside AIS receiver area.

We will assume the following noisy observations:

- `heading intersects area polygon`=1 with a certain probability for all time points, i.e., the vessel is heading towards suspicious area,
- `heading intersects island polygon`=0 with a certain probability for all time points, i.e., the vessel is not heading towards the island,
- `speed larger than zero`=1 with a certain probability for all time points, i.e., the vessel is moving,
- `distance to zone is decreasing`=1 with a certain probability for all time points, i.e., the vessel is approaching AIS area boundary.

We evaluated the DBN with different observation probabilities, i.e., different amount of noise in the observation data. Figure 13a), b), and c) shows the result where the probability of wrong observations is 0.1, 0.3, and 0.5, respectively.In the second scenario, where the vessel is an incoming smuggling vessel, we assume the following deterministic observations:

- `past point in polygon check`=1 for all time points, i.e., the vessel was in a suspicious area,
- `checking object type`=1 for all time points, i.e., the vessel is no tanker/cargo/passenger vessel,
- `MMSI-Number is empty`=1 for all time points, i.e., vessel has an unknown ID,
- `point in polygon check`=0 for all time points, i.e., vessel is not inside AIS receiver area.

We assume the following noisy observations:

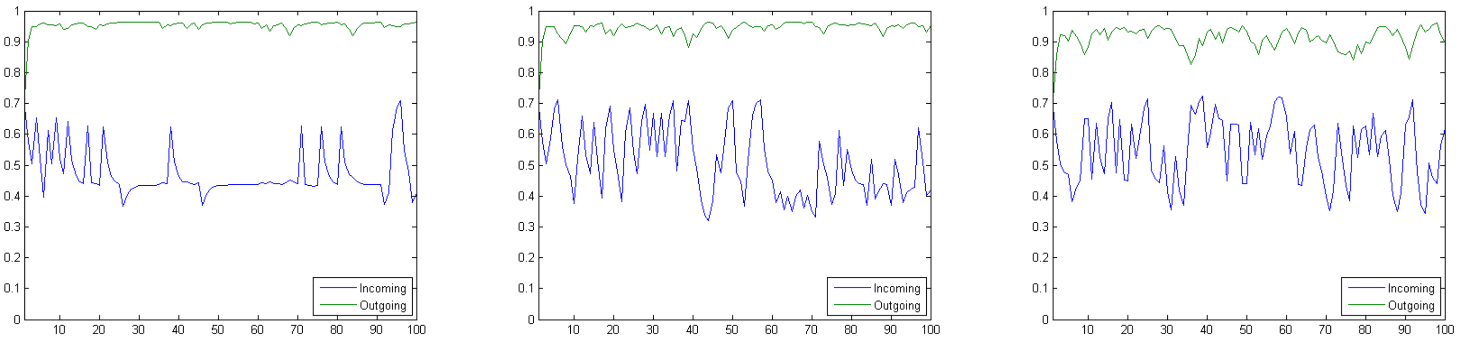
- `heading intersects area polygon`=0 with a certain probability for all time points, i.e., the vessel is not heading towards suspicious area,
- `heading intersects island polygon`=1 with a certain probability for all time points, i.e., the vessel is heading towards the island,
- `speed larger than zero`=1 with a certain probability for all time points, i.e., the vessel is moving,
- `distance to zone is decreasing`=1 with a certain probability for all time points, i.e., the vessel is approaching the boundary of the AIS-area.

Like in the first scenario, we present the results with different observation noise in Figure 14. In both scenarios, we see that the two situations of `suspicious incoming smuggling vessel` and `suspicious outgoing smuggling vessel` can be clearly distinguished from each other, even if we add noisy observations with a probability of 0.5. In both cases, the probability of the underlying situation is around 0.9 or higher, whereas the probability of the other situation is much lower. Of course, if we add more noise, the probability of situation that is not true is more unstable over time. The values range from around 0.4 to 0.7. The reason that the probability is not close to zero is that we have observations that support both situations. Thus, we have contradictory observations for the situation that is not true, which results in probability values between 0.4 and 0.7.

IX. Conclusion and Future Work

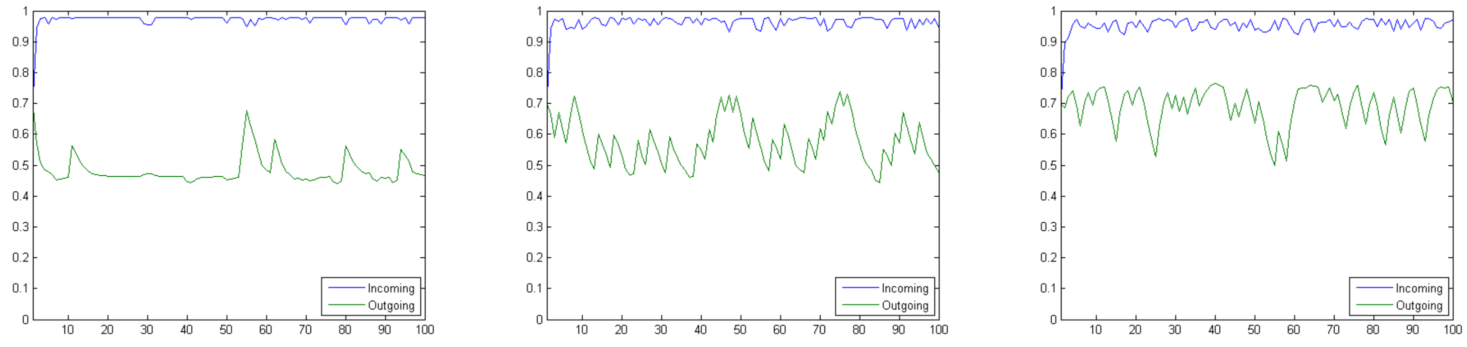
In this article, the information flow in an intelligent surveillance system was highlighted. We described the process of situation assessment in detail and showed how it can be included into the information flow.

The main contribution of this work was the establishment of systematic approach to characterize and recognize situations of interest in a probabilistic way. The focus hereby was on a top-down approach, i.e., that a maritime expert is able to model the situations of interest by necessary and sufficient conditions regarding other situations. The result of the characterization



(a) Probability of wrong observation: 0.1 (b) Probability of wrong observation: 0.3 (c) Probability of wrong observation: 0.5

Fig. 13. Outgoing smuggling vessel scenario: probabilities of incoming and outgoing smuggling vessel over 100 time steps with different observation noise.



(a) Probability of wrong observation: 0.1 (b) Probability of wrong observation: 0.3 (c) Probability of wrong observation: 0.5

Fig. 14. Incoming smuggling vessel scenario: probabilities of incoming and outgoing smuggling vessel over 100 time steps with different observation noise.

process is a Situational Dependency Network (SDN), of which a Dynamic Bayesian Network (DBN) can be generated automatically. For the generation of the DBN, we presented several rules for determining the parameters. During the process of recognizing situations, the DBN uses observation values of so-called elementary situations and is able to determine the probability of more abstract situations over time by using well-known efficient inference methods.

Finally, an application example of a SDN in the maritime domain was given. We generated a DBN by using the established rules and evaluated the DBN by using noisy observations. Especially, we showed that the network behaves as a user would expect. By using this approach, the operator would be able to define situations of interest by himself and to perform a probabilistic situation assessment without the use of training data. Future work includes a refinement of the parameter settings and an evaluation with real data.

ACKNOWLEDGMENT

The work was conducted during a research visit at Ningbo University (China), which was supported by Karlsruhe House of Young Scientists (KHYS).

REFERENCES

[1] Y. Fischer and J. Beyerer, “A top-down-view on intelligent surveillance systems,” in *Proc. of the 7th International Conference on Systems (ICONS)*. IARIA, 2012, pp. 43–48.

[2] M. R. Endsley, “Towards a theory of situation awareness in dynamic systems,” *Human Factors*, vol. 37, no. 11, pp. 32–64, 1995.

[3] Automatic Identification System (AIS) provided by the International Maritime Organization (IMO), last access: 08.12.2013. [Online]. Available: http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, 2012.

[6] M. Andersson and R. Johansson, “Multiple sensor fusion for effective abnormal behaviour detection in counter-piracy operations,” in *Proc. of the 2nd International Conference on Waterside Security*, 2010, pp. 1–7.

[7] D. Meyer-Delius, C. Plageman, and W. Burgard, “Probabilistic situation recognition for vehicular traffic scenarios,” in *Proc. of the IEEE Int. Conference on Robotics and Automation*, 2009, pp. 459–464.

[8] D. L. Hall and S. A. H. McMullen, *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Inc., 2004.

[9] M. Baum, I. Gheta, A. Belkin, J. Beyerer, and U. D. Hanebeck, “Data association in a world model for autonomous systems,” in *Proc. of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2010, pp. 187–192.

[10] A. Dore, M. Soto, and C. S. Regazzoni, “Bayesian tracking for video analytics: An overview,” *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 46–55, 2010.

[11] A. Bauer, T. Emter, H. Vagts, and J. Beyerer, “Object oriented world model for surveillance systems,” in *Future Security: 4th Security Research Conference*. Fraunhofer Press, 2009, pp. 339–345.

[12] A. N. Steinberg, C. L. Bowman, and F. E. White, “Revisions to the JDL data fusion model,” in *Sensor Fusion: Architectures, Algorithms, and Applications, Proceedings of the SPIE Vol. 3719*, 1999, pp. 430–441.

[13] J. Moßgraber, F. Reinert, and H. Vagts, “An architecture for a task-oriented surveillance system: A service- and event-based approach,” in *Fifth International Conference on Systems (ICONS)*, 2010, pp. 146–151.

[14] Y. Fischer and A. Bauer, “Object-oriented sensor data fusion for wide maritime surveillance,” in *Proc. of the 2nd International Conference on Waterside Security (WSS)*, 2010, pp. 1–6.

[15] J. Ye, S. Dobson, and S. McKeever, “Situation identification techniques in pervasive computing: A review,” *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 36 – 66, 2012.

[16] A. Dahlbom, “Petri nets for situation recognition,” Ph.D. dissertation, Örebro University, Sweden, 2011, last access: 08.12.2013. [Online]. Available: http://oru.diva-portal.org/smash/record.jsf?pid=diva2:384609

[17] J. IJsselmuiden, A.-K. Grosselfinger, D. Münch, M. Arens, and R. Stiefelhagen, “Automatic behavior understanding in crisis response control rooms,” in *Proc. of International Joint Conference on Ambient Intelligence*, 2012, pp. 97–112.

[18] M. Kokar and M. Endsley, “Situation awareness and cognitive modeling,” *Intelligent Systems, IEEE*, vol. 27, no. 3, pp. 91 –96, 2012.

[19] M. Kokar, C. Matheus, and K. Baclawski, “Ontology-based situation awareness,” *Information Fusion*, vol. 10, no. 1, pp. 83–98, 2009.

[20] J. Garcia, J. Gomez-Romero, M. Patricio, J. Molina, and G. Rogova, “On the representation and exploitation of context knowledge in a harbor surveillance scenario,” in *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, 2011, pp. 1–8.

[21] A. C. Van den Broek, R. M. Neef, P. Hanckmann, S. P. Van Gosliga, and D. Van Halsema, “Improving maritime situational awareness by fusing sensor information and intelligence,” in *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, 2011, pp. 1–8.

[22] A. Vandecasteele and A. Napoli, “An enhanced spatial reasoning ontology for maritime anomaly detection,” in *7th International Conference on System of Systems Engineering (SoSE)*, 2012, pp. 1–6.

[23] J. Roy and M. Davenport, “Exploitation of maritime domain ontologies for anomaly detection and threat analysis,” in *International Waterside Security Conference (WSS)*, 2010, pp. 1–8.

[24] G. Pilato, A. Augello, M. Missikoff, and F. Taglino, “Integration of

ontologies and bayesian networks for maritime situation awareness," in *IEEE Sixth International Conference on Semantic Computing (ICSC)*, 2012, pp. 170–177.

[25] M. Kruger, L. Ziegler, and K. Heller, "A generic bayesian network for identification and assessment of objects in maritime surveillance," in *15th International Conference on Information Fusion (FUSION)*, 2012, pp. 2309–2316.

[26] W3C Web Ontology Language, last access: 08.12.2013. [Online]. Available: http://www.w3.org/2004/OWL

[27] P. C. G. Costa, "Bayesian semantics for the semantic web," Ph.D. dissertation, School of Information Technology and Engineering, George Mason University. Fairfax, VA, USA, 2005, last access: 08.12.2013. [Online]. Available: http://digilib.gmu.edu/dspace/handle/1920/455

[28] K. B. Laskey, "MEBN: A language for first-order bayesian knowledge bases," *Artificial Intelligence*, vol. 172, no. 2-3, pp. 140–178, 2008.

[29] S. C. Dinkel, W. Hafner, P. C. G. Costa, and S. Mukherjee, "Service-based situational awareness on the semantic web," in *Proc. of the IEEE 3rd Int. Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2013, pp. 249–256.

[30] R. Carvalho, R. Haberlin, P. Costa, K. Laskey, and K. Chang, "Modeling a probabilistic ontology for maritime domain awareness," in *Proc. of the 14th Int. Conference on Information Fusion*, 2011, pp. 1 –8.

[31] K. B. Laskey, R. Haberlin, R. N. Carvalho, and P. C. G. Costa, "PR-OWL 2 case study: A maritime domain probabilistic ontology," in *Proceedings of the 6th International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS 2011).*, 2011, pp. 76–83.

[32] Y. Fischer, A. Bauer, and J. Beyerer, "A conceptual framework for automatic situation assessment," in *Proc. of the IEEE 1st International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2011, pp. 234–239.

[33] Y. Fischer and J. Beyerer, "Acceptance of automatic situation assessment in surveillance systems," in *Proc. of the IEEE 2nd International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012, pp. 324–331.

[34] Y. Fischer and J. Beyerer, "Defining dynamic bayesian networks for probabilistic situation assessment," in *Proc. of the 15th International Conference on Information Fusion*, 2012, pp. 888–895.

[35] Y. Fischer and J. Beyerer, "Ontologies for probabilistic situation assessment in the maritime domain," in *Proc. of the IEEE 3rd International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, San Diego, USA, feb 2013, pp. 105–108.

[36] K. P. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, Berkeley, 2002, last access: 08.12.2013. [Online]. Available: http://www.cs.ubc.ca/˜murphyk/Thesis/thesis.html

[37] GeNIe & SMILE, last access: 08.12.2013. [Online]. Available: http://genie.sis.pitt.edu/

[38] Bayes Net Toolbox by Kevin Murphy, last access: 08.12.2013. [Online]. Available: https://code.google.com/p/bnt/

[39] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Urban Area Energy Flow Microsimulation for Planning Support: a Calibration and Verification Study

Diane Perez, Jérôme Henri Kämpf, and Jean-Louis Scartezzini
Solar Energy and Building Physics Laboratory (LESO-PB)
École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
{diane.perez; jerome.kaempf; jean-louis.scartezzini}@epfl.ch

***Abstract*—A recently developed energy management platform named MEU (an acronym for Management of Energy in Urban areas) has been tested on three case study urban areas of a few hundred buildings in the Swiss cities of la Chaux-de-Fonds, Neuchâtel and Martigny. The MEU simulation framework for energy-efficient systems simulates the buildings' energy demand, infers the production of the connected energy conversion systems, and simulates the complete demand and supply energy flow picture. The platform is designed to use monitored consumption data if available, and, where it is not, intends to produce correct estimates of the average energy demands of housing and administrative buildings. This article explores in detail the capacity of the platform to correctly represent existing urban areas' energy flow based on the limited data available by comparing the simulated values with monitored data. Default values are carefully chosen to obtain a statistically adequate match, thus strengthening the confidence in simulation results when no monitored data is available. The study also leads to interesting observations and hypotheses regarding the energy efficiency of existing buildings, and provides useful conclusions about the possibilities and limitations of the simulation of disaggregated urban energy flow.**

Keywords*—urban energy flow simulation, heating demand simulation, calibration, verification, monitored energy consumption, energy demand and supply.

I. Introduction

DESPITE raising awareness of the problems associated with the widely unsustainable modern energy use, much improvement is still needed to reduce resource consumption. A large part of energy use can be attributed to buildings in urban zones, and urban populations are increasing. It is thus of interest for research to propose innovative energy efficiency measures and energy management tools for the urban context. Such a tool, the MEU platform, is described and tested in detail in this paper, after the first results were presented in [1].

The MEU platform was developed through a collaboration between several Swiss research labs (CREM in Martigny, Hes-so Valais, CEN, LENI and LESO-PB at EPFL), cities (Lausanne, La Chaux-de-Fonds, Martigny and Neuchâtel) and energy utilities (Synergy, Viteos, SSIGE). Its objective is to offer a platform for the management of energy in urban areas, considering neighbourhoods of a few hundred buildings and based on the limited available data. Among the functionalities required, specific features of this platform include:

- consideration of individual buildings,
- modelling of supply as well as demand,
- combined use of available monitored data and specialised simulation tools to produce a coherent picture of the urban energy flow,
- parallel modelling of the existing situation and of energy efficiency scenarios.

The third point in particular is, to our knowledge, an uncommon feature. A specific approach is required to deal with conflicting simulation results and monitored values. On the other hand, the procedure offers a valuable opportunity to study how micro-simulation tools for energy demand and supply can be used to estimate real urban energy consumptions. The study presented here exploits this opportunity as part of the verification and calibration process of the MEU platform.

The next section of this article reviews related research in the domain of urban energy modelling. Section III gives an overview of the methodology of the MEU platform, covering its modelling approach, simulation method and implementation structure. Section IV describes the models of the case study neighbourhoods, the data sources used to create the models and the choice of default values. Section V presents the results of the calibration and comparisons between simulated and monitored values. The results obtained through simulations are discussed in Section VI, and Section VII provides a concise conclusion of this paper.

II. State of the art

Among the large number of research domains concerned by urban energy simulation [2] [3], there is now a growing interest in the evaluation of the energy need of larger and/or pre-existing urban areas [4], to evaluate the energy performance associated with alternative development or improvement scenarios. However, it remains a challenge today to simulate the detailed energy flow at the scale of a few hundred buildings, including the demand and the supply sides [5]. The advantage of this simulation approach is that it allows for the test of scenarios of various levels of detail, covering in a large part the options available to local politicians and energy departments. The simulation of existing buildings' disaggregated energy flow at a large scale is however complicated by a lack of information regarding the buildings' characteristics and energy supply situation, and a limited number of verification studies comparing simulation tools' results and monitored data.

Even when simulating individual new buildings, where construction characteristics are well known, the simulated energy demands and real consumption can differ significantly [6]. Regarding existing buildings, a successful method is to use well-calibrated statistical models, for which aggregated results match monitored data quite well [7]. Nevertheless, more detailed data sources exist: cadaster and building geometry data, building registers, monitored consumption data, other geographical information system (GIS) data, etc. Together with the development of decentralised energy production systems and the need for more localised information, this supports the development of disaggregated urban energy simulation models [8]. An interesting verification study at this level was performed on a 700-building area in Germany [9], evidencing an average dispersion between simulated and monitored consumption of individual buildings above 30%, while the overall annual consumption was reasonably well estimated.

Two factors explain the difficulty to simulate individual buildings at a large scale: the limited level of detail of data available for such simulation, and the stochastic behaviour of occupants. Nevertheless, acknowledging our inability to model all parameters does not decrease the benefits of using all available data: a disaggregated model making an intelligent use of default or standard data to mitigate unavailable data has the potential to provide better results than statistical models, without limiting possible improvements of the model. The calibration of the model and verification of default values, which is the primary focus of this article, is however of uttermost importance.

As discussed in [10], the amount of data (whether real or default) involved in this kind of study, as well as its quality and longevity are important concerns. However, few publications in the domain tackle this problem. The MEU platform presented in this study was developed with this concern in mind. It was designed to provide the simulation functionalities necessary for the energy management of urban zones, accounting for both demand and supply sides, whilst addressing some of the data concerns related to this domain.

III. Methodology

The simulations presented here were performed with the MEU platform, an urban energy management tool developed these last four years through a collaboration between research units, energy utilities and municipalities. Not just an energy demand simulation tool, the MEU platform intends to manage energy-related data in an adapted data model, represent the demand and supply energy picture, offer a structure for a combined use of monitored data and simulation tools, and integrate standard analysis functionalities.

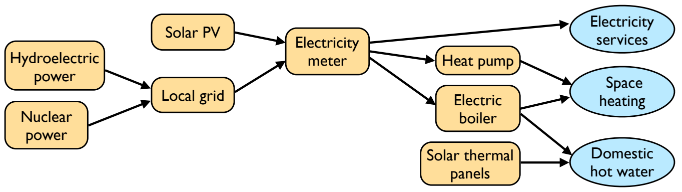


Figure 1. Example graph representation of the energy flow providing a building's energy services.

A. Energy system model

The MEU platform models the urban energy flow as an oriented graph, with the following rules: *source* nodes (in a broad sense) are linked to (i.e., provide energy to) *energy conversion system (ECS)* nodes or *network* nodes. The network nodes themselves are linked to other network or ECS nodes, and the ECS nodes in turn are linked to other ECS nodes or *building* nodes, seen as energy sink nodes (Figure 1).

This graph approach was chosen for two main reasons: without it, even simple situations where a building's domestic hot water (DHW) is provided by solar thermal panels completed with an electrical boiler while space heating is supplied by an oil boiler soon become very complicated to model. Furthermore, the intent to consider monitored data highlights its natural attribution to ECS instead of buildings: a gas consumption usually corresponds to a gas boiler, whether this boiler provides only one energy service in one building or several services in a group of buildings. In the MEU graph model, monitored energy consumptions can be attributed to any ECS node as its input flow.

Energy demands for space heating, DHW, electricity and cooling services are associated to each building node. An arbitrary number of connections link each of these demands with the ECS nodes supplying them. Each connection records information about which fraction of the demand is supplied through this connection.

The source nodes actually represent energy in any form that does not need to be investigated further: it can be natural gas as delivered in the country, or a pre-defined standard electricity mix. These nodes are defined along with their name by environmental factors, such as a *kWh primary energy per kWh* coefficient and a *kg* $CO_{2,eq}$ *per kWh* coefficient, based on the EcoInvent database [11], which intend to unite such sustainability data in one coherent data bank.

Network nodes are mainly characterised by a loss factor, whereas ECS nodes refer to a dedicated black-box simulation model including any number of modifiable parameters. Building nodes include data about address, location (footprint), allocation and a physical model used for the estimation of heating and cooling demand with the simulation program CitySim presented below. Electricity and DHW needs are estimated using Swiss norms [12], based on main allocation and treated floor area.

B. Energy flow simulation

The simulation process consists of three main steps:

- The whole annual energy flows are first estimated, creating a complete but fully-simulated energy flow picture. The energy service demands of each building are simulated by a dedicated module: the CitySim web service. The graph model structure then ensures sufficient information is available to resolve the energy flow providing those demands. The ECS nodes losses are simulated using the corresponding black-box models, offering flexibility regarding the available ECS models.
- The simulated energy flow picture is then adapted (scaled) to match the monitored consumption values available. The intent is to create an energy flow picture as close to reality as possible, by combining the incomplete information of monitored data with the structuring simulation results. Both original *simulated values* and subsequently *adapted values* are saved for later analyses.
- The last step consists in retrieving usually required results from the fully-informed energy flow picture, including building-based values of primary, final (delivered) and useful energy use per service (as defined in [13]). The results produced also include map representations of the relative energy efficiency of buildings, as well as overall results, such as the relative shares of energy carrier used or the renewable fraction of the primary energy consumed.

The first results presented in [1] confirmed that the approach described above could correctly represent urban energy flow. The test scene included several cases of centralised ECS providing space heating and / or DHW in different buildings, buildings where space heating is produced by both the district heating network and a gas boiler (in order to free power on the district heating network during heavy load periods), and electricity meters providing both the electricity demand and an electrical boiler, which were correctly simulated by the MEU platform.

On top of the pre-computed results, the energy flow picture obtained contains a large amount of information that can be accessed to perform more detailed analyses. In particular, the co-existence of the simulated and adapted values provides a valuable tool for the analysis of the validity of the model. We will consider the discrepancy of those two values at the level of the buildings' energy demands, using a discrepancy factor f_2 defined as the logarithm in base 2 of their ratio:

$$f_2 = \log_2\left(\frac{\text{simulated value}}{\text{adapted value}}\right) \tag{1}$$

The choice of this indicator comes from two observations: firstly, when dealing with a large number of buildings, the meaning of a particular monitored consumption value is often uncertain, i.e., it is not always well defined which services and buildings are concerned. As such, using the monitored values as reference was not deemed a reliable method, no more than using the simulated values as reference. Secondly, using the percentage of deviation from a reference value has the drawback of not being a symmetrical indicator: a 50% result can be considered as an equivalent error to a 200% result. The use of the f_2 factor avoids such problems with symmetric values around zero (which corresponds to a perfect match). An f_2 value of -1 corresponds to a simulated value twice smaller than the adapted value, and an f_2 value of 1 corresponds to a simulated value two times higher than the adapted value, without any hypothesis regarding which is more reliable. A map representation of this indicator is also very useful to quickly spot locations where simulated values and monitored data do not match, in order to verify and correct the underlying model.

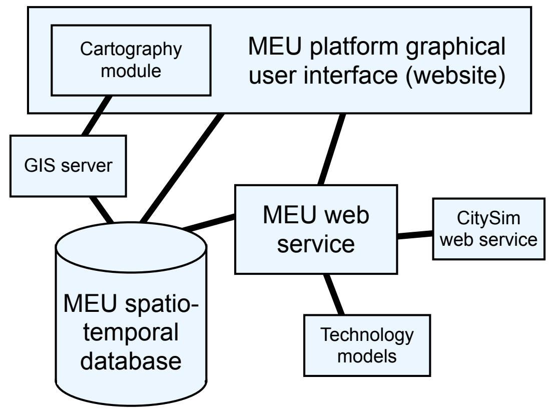


Figure 2. Web-based structure of the MEU platform.

C. The MEU platform

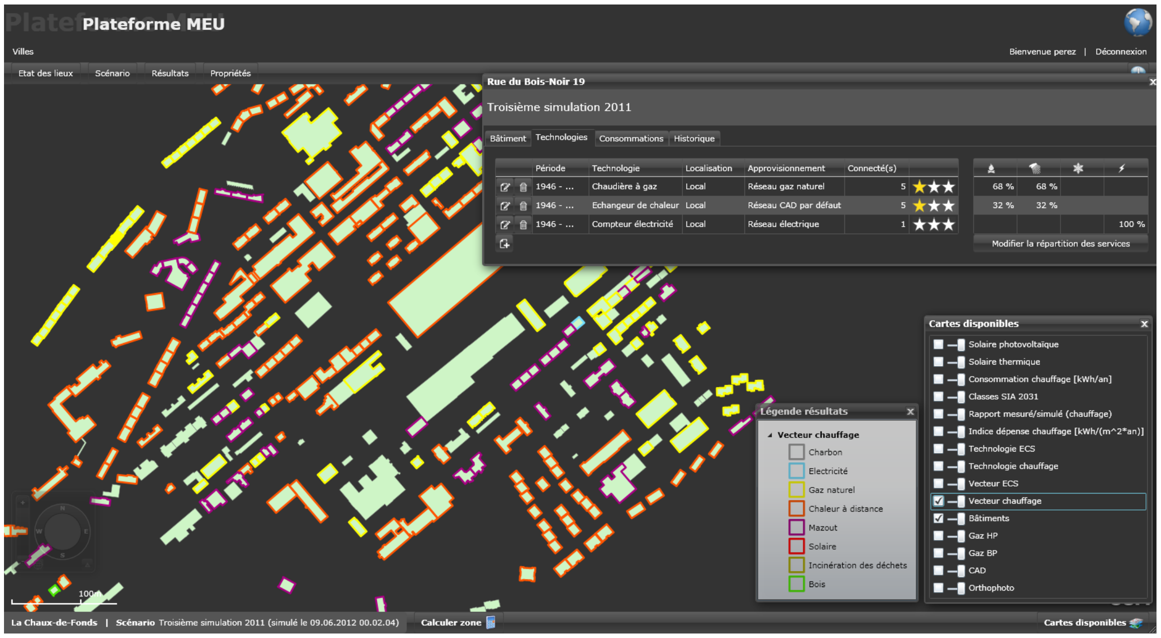
The MEU simulation framework was implemented as a web-based platform with decentralised web services (Figure 2). A GIS-based web interface provides access and editing functionalities to the model through a map representation (Figure 3). It also grants access to the simulation results, in the form of map representations as well as numerical results, for individual buildings or aggregated for the whole scene.

The data is stored in a spatio-temporal PostgreSQL (open-source) database and can also be accessed directly. The simulation of the energy flow is implemented in the dedicated MEU web service. During the simulation process the MEU web service connects to the database to retrieve the necessary input data and later to save results, calls the CitySim web service for the estimation of energy demands, and uses the "technology models" module to simulates the production of an ECS based on its consumption or vice-versa.

D. CitySim

The urban energy use simulator CitySim [14] was developed at EPFL based on multiple physical models coupled together. Its simulation results have been compared against ESP-r, EnergyPlus and the European norm CEN 13790, demonstrating similar results [14]–[16]. The verification study presented here concerns to a large extent the simulation of heating needs by the CitySim web service, which is described in some detail.

CitySim can compute an estimation of the on-site energy use for heating, cooling and lighting with an hourly time step. A radiation model first computes the irradiation incident on



Période	Technologie	Localisation	Approvisionnement	Connecté(s)		🔥	🚿	❄	⚡
1946 - ...	Chaudière à gaz	Local	Réseau gaz naturel	5	★★★	68 %	68 %		
1946 - ...	Echangeur de chaleur	Local	Réseau CAD par défaut	5	★★★	32 %	32 %		
1946 - ...	Compteur électricité	Local	Réseau électrique	1	★★★				100 %

Vecteur chauffage
- ▢ Charbon
- ▢ Electricité
- ▢ Gaz naturel
- ▢ Chaleur à distance
- ▢ Mazout
- ▢ Solaire
- ▢ Incinération des déchets
- ▢ Bois

- ☐ Solaire photovoltaïque
- ☐ Solaire thermique
- ☐ Consommation chauffage [kWh/an]
- ☐ Classes SIA 2031
- ☐ Rapport mesuré/simulé (chauffage)
- ☐ Indice dépense chauffage [kWh/(m^2*an)]
- ☐ Technologie ECS
- ☐ Technologie chauffage
- ☐ Vecteur ECS
- ☑ Vecteur chauffage
- ☑ Bâtiments
- ☐ Gaz HP
- ☐ Gaz BP
- ☐ CAD
- ☐ Orthophoto

Figure 3. The web interface of the MEU platform, here showing the main energy carrier used for heating in each building and the possibility to define any number of energy conversion systems to provide the energy services.

each surface of the scene, direct from the sun, diffuse from the sky and reflected by other surfaces. The results of this model, together with predictions of long-wave radiation exchange, are input to a thermal model. This model determines the thermal exchange through building envelopes and computes the heating and cooling energy needs to maintain predefined temperature conditions inside. Finally, ECS providing heating, cooling and electricity can also be defined.

As input, a complete physical description of the scene as well as climatic data are needed for the simulation. The climatic data includes hourly temperature, wind and irradiance values, together with the geographic coordinates and the definition of far field obstructions (which is used by the radiation model). The building models describe the envelope of each building (the thermal properties of each facade, the layered composition of the walls with thermal inertia and transmittance properties, the proportions of window openings and the physical properties of the glazing) as well as the infiltration rate and the presence of occupants and heat gains.

CitySim is used in the MEU platform to compute heating and cooling demands only. It was transformed for this purpose into a web service, also including the estimation of electricity and DHW annual demands based on the SIA norms [12]. The central MEU web service prepares the input model based on the data available in the database, calls the CitySim web service and retrieves the simulated energy demands to save them in the database.

IV. Case study model

The verification study considers three case study urban areas, located in the Swiss cities of Neuchâtel, La Chaux-de-Fonds and Martigny. Still in its verification phase, the platform has, to date, not been used in other countries. This section describes the data sources used to create the models, and gives some specific insights on each of the urban zones. The default data chosen to complete these models is discussed in the next section.

A. Data sources

The models created for this project are based on cadastral data defining buildings' footprints, and possibly their type (allocation). The footprints are combined with data from the national building register including the address, period of construction, number of floors and optionally space heating and DHW supply systems. These were completed with a large amount of default data to form the physical model of the building (see Section IV-C).

The monitored consumption of electricity, gas and heat from the district heating network (DHN) were obtained through the local energy providers. Part of the fuel oil (supplied by various companies) consumption values were provided by the cities, based on contacts with building owners. The model is completed with locally measured meteorological data for the year corresponding to monitored consumption data.

The concerned municipalities' help permitted to slightly simplify the collection of data. Nevertheless, the combination of the various sources into a coherent model remained a time-consuming task, accounting for several weeks of work for each case-study model.

B. Case study urban zones

The three cities are located in the western part of Switzerland, but cover quite different climatic conditions. Figure 4 shows the buildings' footprints and their construction period.

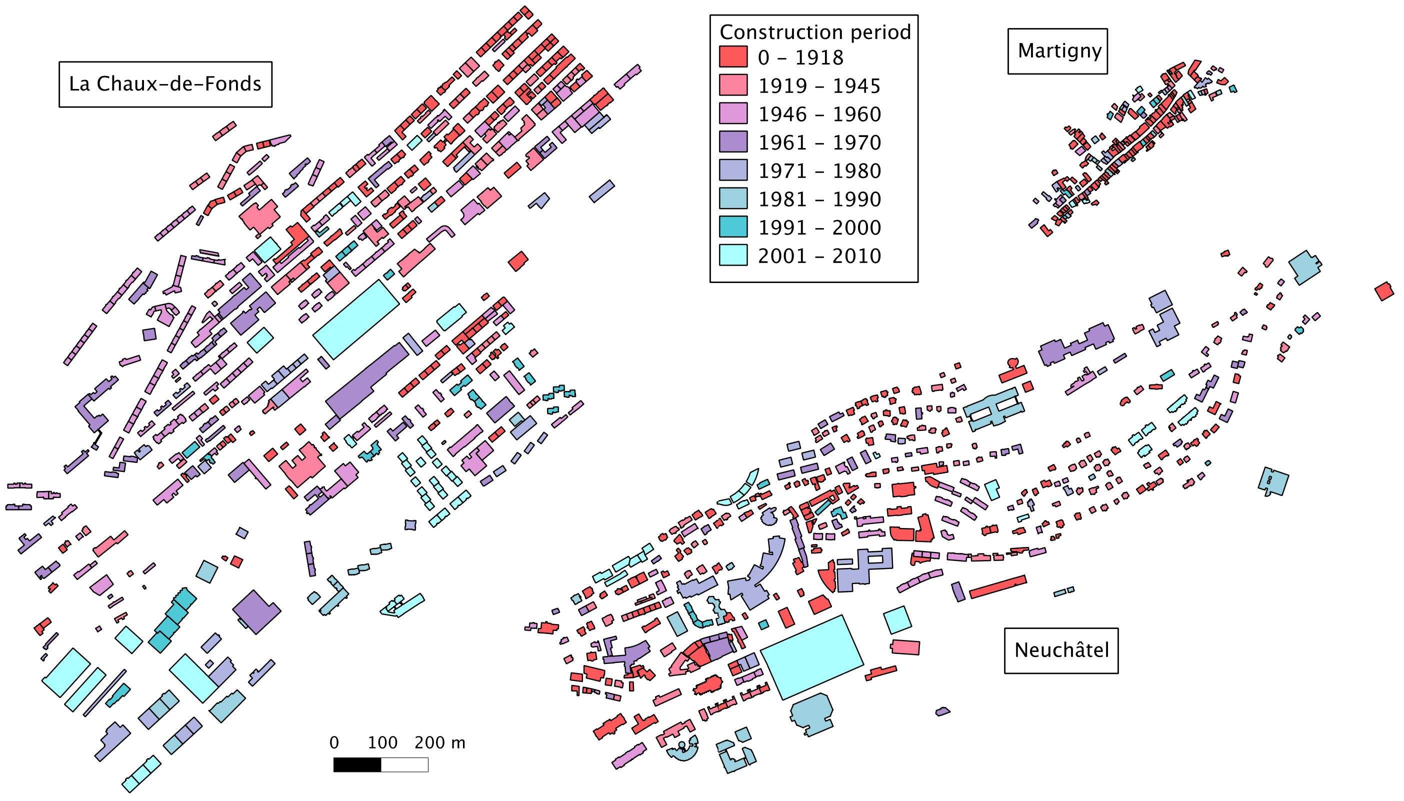


Figure 4. Maps of the three case study areas, showing the buildings' construction period (colours available online).

1) La Chaux-de-Fonds (CdF): Located in the Jura sub-alpine mountain range, La Chaux-de-Fonds (alt. 1000m) is a UNESCO World Heritage Site for its watchmaking industry-driven urbanism mixing housing and workshop at the heart of the city. The case study area covers the main part of the city center and is composed of 600 buildings. It includes mostly multi-family houses and industrial or workshop buildings, heated through a district heating network (DHN), gas or oil.

The model was up to some extent verified with the help of the energy office and energy supplier of La Chaux-de-Fonds, producing the highest confidence level case study model. Energy consumption data for the year 2011 (3853 degree-days[1]) was provided for electricity, gas and the DHN as well as for oil, although part of this data was extrapolated from other years.

2) Neuchâtel (Nch): Neuchâtel is located in the Swiss plateau, between the Alp and the Jura mountains. The case study area is a part of the city center, covering a slope between a lake and the first shoulders of Jura (alt. 430m-500m), and composed of approx. 400 single-family houses, multi-family houses, commercial buildings and other types of buildings. This most heterogeneous case study area is also heated with gas, oil and DHN.

Digital surface and terrain models (DSM and DTM) were available for the creation of the model, providing individual building's average altitude and height. The model was less intensively checked than that of CdF, but can rely on monitored gas and DHN consumption data for the year 2008 (3166 DD).

3) Martigny (Mrt): Martigny (470m) is located in the Rhone valley in the western part of the Alps range; the case study area is a very compact housing neighbourhood of approximately 200 buildings west of the city center. Unlike the two other case study areas, a large share of the buildings are heated with electricity, the others using mostly oil or gas. Monitored consumption values are available for 2010 (3116 DD). The model is however the least verified, and an unknown part of the buildings might use a wood stove for complementary space heating. The case study was still included in our analysis for representativeness, although the results obtained with it were considered with a lower weight.

C. Default data

As mentioned above, in order to obtain a microsimulation model at this scale of a few hundred buildings, numerous default values and rules were used. Only a very limited amount of data was considered as compulsory to create the case study models: the buildings' footprint, period of construction, main allocation (type) and number of floors (although the use of more available data was always possible).

For this verification study, default values were first chosen based on the information available to us, through published

[1]Using $\bar{T}_d$ the average temperature of day d, the degree-days (DD) were computed as $DD = \sum_{d=1}^{365} \left\{ 0 \text{ if } \bar{T}_d > 12; \quad 20 - \bar{T}_d \text{ if } \bar{T}_d \leq 12 \right\}$

Table I. DEFAULT PHYSICAL PROPERTIES OF THE BUILDINGS: VENTILATION RATE n_{VENT} [H^{-1}], WALL U-VALUE U_w [W/M^{2}K], ROOF U-VALUE U_r [W/M^{2}K] AND GROUND K-VALUE K_g [W/M^{2}K]. DEFAULT WALL TYPES ARE DESCRIBED OUTSIDE TO INSIDE, THE VARIOUS VERSION BEING SLIGHTLY BETTER OR LESS INSULATED.

Period	Wall description	Version 1				Version 2				Version 3			
		U_w	U_r	K_g	n_{vent}	U_w	U_r	K_g	n_{vent}	U_w	U_r	K_g	n_{vent}
Before 1918	Rough-stone wall	1.41	1.9	2.8	0.70	0.90	0.70	1.4	0.60	0.94	0.50	1.0	0.60
1919 - 1945	Rough-stone wall	1.41	1.9	2.8	0.70	0.90	0.70	1.4	0.60	0.94	0.40	0.9	0.60
1946 - 1960	Rough-stone, air gap, brick	1.35	1.4	2.3	0.60	0.98	0.70	1.5	0.60	1.35	0.85	1.5	0.75
1961 - 1970	Brick, air gap, brick	1.14	1.3	2.0	0.55	0.91	0.65	1.3	0.55	1.03	0.70	1.3	0.70
1971 - 1980	Brick, insulation, brick	0.58	0.70	1.3	0.50	0.67	0.60	1.1	0.50	0.86	0.70	1.2	0.65
1981 - 1990	Ins., armed concrete	0.42	0.40	0.63	0.40	0.62	0.43	0.68	0.45	0.90	0.65	1.0	0.60
1991 - 2000	Ins., armed concrete	0.29	0.28	0.42	0.35	0.44	0.31	0.49	0.40	0.69	0.55	0.85	0.55
2001 - 2010	Ins., armed concrete	0.21	0.20	0.28	0.30	0.36	0.25	0.35	0.40	0.51	0.45	0.70	0.55

work as well as surveys or informal knowledge transmission. Given the limited available knowledge regarding the existing building stock's physical properties and energy supply situation, the first version (Version 1) of the model was not expected to provide a good match with monitored data, but rather a basis for the definition of more adapted but still realistic default values. The objective of such a crude model is not to obtain precise individual building energy demands, but representative average results.

Regarding the simulation of energy demand, energy consumption studies show that the most influent parameters are the dimensions of the building, its age and its type (allocation) [17], [18]. The dimensions of the buildings are obtained through their footprints and number of floors. The default values are thus attributed based on the buildings' period of construction and type. Sensitivity analyses performed with CitySim, in accordance with [19], show that after dimension-related parameters, the most influent parameters for the simulation of heating loads are the set point temperature, the ventilation rate and the insulation thickness (or more generally the outer surfaces' properties), followed by internal heat gains and climatic conditions.

The closest measured climatic data was obtained through a national database. Swiss norms recommend the use of a heating set point temperature of 21 °C for housing and administrative buildings' thermal simulations. This important parameter can vary considerably depending on the occupants preferences, but cannot be refined based on the data available to us. Together with electricity and DHW needs, internal heat gains are estimated based on norms.

As for dimension-related parameters, the simulation uses a 3D flat-roof model based on the cadastral footprint of the buildings and their number of floors. The fractions of façades that are shared between heated buildings are considered to be adiabatic. The DSM and DTM data of Neuchâtel's case study corresponds to an average height of 2.73 m per floor as recorded in the register of building. This value is used to estimate the unknown heights of buildings, while a treated floor area to footprint ratio of 0.8 per floor is assumed. The heated volume is further estimated considering 20 cm thick slabs and 10% of the volume occupied by furnitures.

The ventilation rate and construction properties are more uncertain and will thus be the focus of our analysis. The ventilation rate parameter represents the building's air volume change per hour in usual conditions. It strongly depends on the air-tightness of the envelope, the existence of an HVAC (heating, ventilation and air conditioning) system and the occupants' stochastic behaviour regarding ventilation (window opening to avoid overheating is considered separately in CitySim simulations). Although one of the most influential parameters for the simulation, it is thus one of the least accessible. Studies on the air tightness of buildings present a large range of results, corresponding to infiltration rates ranging from 0.1 to 1 h^{-1} [20]–[22]. We thus chose default values for the ventilation rates based on two considerations: most studies show that the infiltration rate of older buildings is 2 to 4 times higher than that of more recent buildings, and Swiss norms recommend a minimum total ventilation rate of 0.3 h^{-1} for comfort and health purposes. The original default values are shown in the "Version 1" part of Table I (the versions 2 and 3 are discussed further on).

Table II. DEFAULT WINDOW TO WALL RATIO α_{win} [-], WINDOW AREA U-VALUE [W/M^{2}K] AND WINDOW AREA G-VALUE [-], BASED ON THE AVERAGE OBSERVED WINDOWS PROPERTIES. THE VALUES CONCERN THE FULL WINDOW AREA, INCLUDING AN AVERAGE OF 25% OF FRAME.

Period	α_{win}	Version 1		Version 2 & 3	
		U_{win}	g_{win}	U_{win}	g_{win}
Before 2000	0.25	2.3	0.47	2.0	0.5
2000 - 2010	0.35	1.3	0.49	1.7	0.5

Regarding the physical properties of the envelope, the construction default parameters must represent the average state of all buildings of each construction period, as the available data does not include information about past thermal retrofitting of buildings. Further, the unknown existence of non-heated attics or cellars complicates the estimation of their thermal resistance (ground K-value and roof U-value). Nevertheless, a first version of default values was based on typical period-specific construction characteristics determined with the help of experimented local architects, and are also shown in Table I.

Windows ratio, U-value and g-value, estimated based on a visual survey of approximately 500 buildings in Zürich, are shown shown in Table II. The calculated U-value and g-value also depend on the hypotheses made regarding the typical glazing properties.

Considering the supply side, wherever monitored consumptions of gas, fuel oil or district heating are available, gas boilers, fuel oil boilers and heat exchangers respectively are defined in the corresponding buildings. It is assumed that these produce space heating, as well as DHW if the building register announced the same energy carrier for both services. The consumption values are then affected to these systems. This

Table III. DEFAULT ENERGY CONVERSION SYSTEMS EFFICIENCY η.

Technology	Version 1 & 2	Version 3
	η [-]	η [-]
Heat exchanger	0.93	0.97
Gas boiler	0.85	0.79
Oil boiler	0.85	0.77
Electric boiler	0.93	0.93
Wood boiler	0.65	0.65
Heat pump (COP)	3.4	3.1
Electricity meter	1.0	1.0

method showed its limits as numerous buildings of Neuchâtel appear to use gas only for cooking, their consumption being clearly incompatible with either space heating or DHW demands.

The lower confidence data of the building register was used to complete the supply picture with other ECS for both space heating and DHW when information was available for buildings without consumption data. At this point, using maps of the f_2 factor, buildings without monitored data for heating that are semi-detached from buildings with a high consumption value were considered to be heated by the same centralised ECS and thus connected to that ECS. Fuel oil boilers were eventually defined in buildings without any other ECS.

An electricity meter providing electrical services was also defined in each building, and associated with the electricity consumption obtained through the energy provider. The electricity consumption of electrical boilers was assumed to be included in the total monitored electricity consumption.

The technology models used to simulate the ECS are quite simple and use the default efficiencies shown in Table III, based on the Swiss norm SIA 2031 [23]. When two or more ECS are defined to provide the same service, it is supposed that each meets the same share of the demand, except for solar thermal panels that are often sized to provide approximately 65% of the annual DHW demand. These default shares can be adapted during the simulation based on the monitored data.

V. DEFAULT DATA CALIBRATION AND MODEL VERIFICATION

The calibration and verification of the model focused on the housing and administrative buildings, which seem the most predictable building types. Other types of buildings are expected to have more varying energy demands, which are barely correlated to the rough data at our disposal. The limited number of such buildings in our case studies also limits the possibilities to perform statistically relevant analyses.

Each case study model was simulated with the MEU platform, first using the Version 1 default values. This section analyses the results using the f_2 factor. Representing the discrepancy between simulated values and monitored data, it provides insights regarding the default values' adequacy as well as indications on other possible model errors. Two improved default values versions were then defined, simulated and analysed.

A. Version 1 simulations

Most of the least reliable default values concern the space heating demand simulation and are attributed based on the construction period. The logarithmic discrepancy factor f_2 for the heating demand is plotted against those periods in Figure 5 for all housing or administrative buildings of each case study.

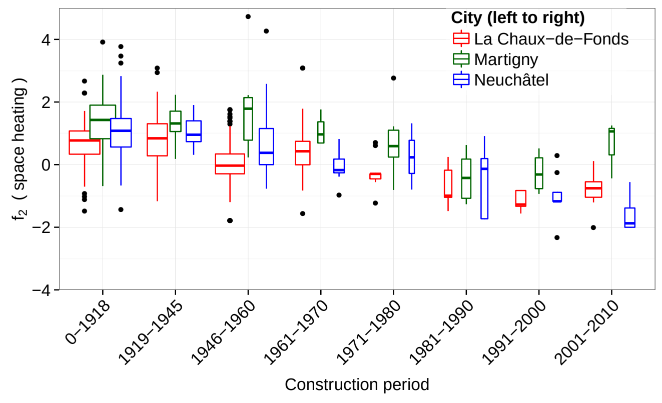


Figure 5. Discrepancy factor of the heating demand as a function of the construction period for the Version 1 simulations. The width of the boxplots is proportional to the square root of the number of observations. Positive f_2 values corresponds to simulated values greater than monitored values, and vice-versa. A zero f_2 value represents a perfect match between simulated and monitored values. Four points outside the range of the graph were ignored. (Colours available online.)

First of all, it must be noted that the largest proportion of buildings in our case studies were built before 1960 (Table IV); the results concerning more recent buildings are less reliable as a result of their limited number. The dispersion of the discrepancy factor is quite high, with a few buildings for which the simulated value was more than 4 times smaller or 16 times greater than the monitored value. However, the interquartile range is between 1 and 2 units of the f_2 factor's scale, which corresponds to ratios of 1:2 to 1:4.

More interesting at this stage is the average value of the f_2 factor, showing that our model globally overestimated the space heating demand of older buildings and underestimated that of more recent constructions. In other words, the thermal efficiency of old buildings was underestimated, while that of recent buildings was overestimated. Surprisingly, the energy use per square meter for heating, represented in Figure 12 and discussed later in Section VI, does not evidence a decrease of the energy consumption with time, except for buildings built after 2000. This goes against our first choice of default values, which supposed a decreasing thermal efficiency for old buildings.

In the case of Mrt, either the heating demand was even more generally overestimated, or the monitored consumption values used for the comparison are too low, which would be coherent with the existence of a non-negligible number of unmodeled wood stoves, the consumption of which could not be taken into account.

B. Version 2 simulations

Based on the previous observations, a second set of default values (Version 2) was defined. The life-time of windows being considerably lower than that of buildings, it was estimated that

Table IV. NUMBER OF BUILDINGS OF HOUSING OR ADMINISTRATIVE TYPE WITH MONITORED HEATING CONSUMPTION. "ALL" GIVES THE TOTAL NUMBER OF BUILDINGS OF HOUSING OR ADMINISTRATIVE TYPE.

Period	CdF	Nch	Mrt
Before 1918	102	47	72
1919 - 1945	44	24	13
1946 - 1960	92	22	8
1961 - 1970	23	12	4
1971 - 1980	13	3	12
1981 - 1990	6	4	9
1991 - 2000	11	8	6
2001 - 2010	29	11	3
Total	320	131	127
All	411	338	155

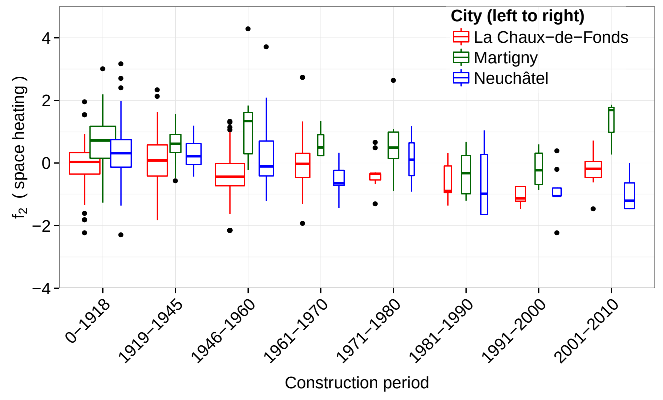


Figure 6. Discrepancy factor of the heating demand as a function of the construction period for the Version 2 simulations. Four points outside the range of the graph were ignored.

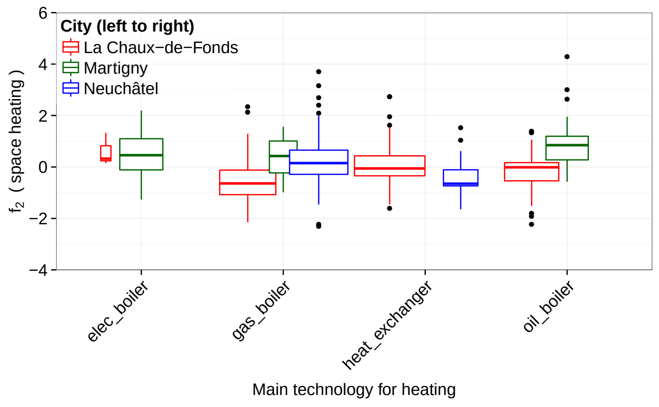


Figure 7. Discrepancy factor of the heating demand for the Version 2 simulations, per technology used for space heating. Four points outside the range of the graph were ignored.

Table V. AVERAGE ANNUAL EFFICIENCIES OF HEAT PRODUCTION FOR EXISTING PLANTS IN 2005 ACCORDING TO TWO STUDIES REGARDING THE SWISS BUILDING STOCK.

Technology	[27]	[28]
Gas	0.79	0.79
Fuel oil	0.77	0.77
Wood	0.64	0.64
Heat pump (COP)	3.3	2.7

the quality of glazing and frame materials was more uniform than previously estimated (Table II). The overall quality of the envelope was revised, based on the hypothesis that the simplest insulation measures for old buildings, and thus the most likely to be widespread, concern primarily the roof and ground. The ventilation rate of old buildings was also reduced, keeping in mind the following remarks:

- To our knowledge, no study regarding air-tightness or ventilation rates of Swiss buildings is available.
- Measurements usually concern the air-tightness of the envelope; the estimation of an average ventilation rate based on those measurements still involves numerous hypotheses (among other regarding wind conditions) that were not taken into account in this work.
- For very leaky buildings, improving the air tightness is possibly easier to accomplish than other energy-efficiency measures.

Conversely, the overall quality of more recent buildings was slightly reduced, among other by diminishing the estimated insulation thickness.

The results of the second simulation, shown in Figure 6, slightly improved the match between simulated and monitored values, but the trends observed in the first simulation remain.

At this point, the correlation between the discrepancy factor and the technology used for heating was also investigated, but no significant trend could be observed (Figure 7). Nevertheless, and although information regarding the average efficiency of existing energy conversion systems is very scarce, the default efficiencies of the heat exchangers, oil boilers and gas boilers were modified the Version 3 to better represent the average quality of installed ECS. Regarding the DHN, according to a local specialist heat exchanger efficiencies are currently of the order of 99%. This value was only slightly decreased to 97% to account for heat losses after the heat exchanger. Losses occurring during heat production and in the distribution network are considered elsewhere, i.e., at the corresponding nodes in the graph model, and were also set according to the local energy provider's data. By contrast, the original hypotheses regarding gas and oil boilers efficiencies were probably too optimistic, as they correspond to values given by the norm [23] for correctly sized condensing boilers. Regarding conventional boilers, the norm proposes an annual efficiency of 80%, and less in case of bad sizing. An official document regarding the sizing of boilers also mentions typical annual efficiencies of 70 to 85% [24]. Globally, the overall space heat production efficiency in Switzerland is estimated at 78% by [25]. Moreover, the efficiencies are traditionally computed based on the lower calorific value of the substance in Europe [26] (the convention used by the documents cited above is unspecified), while the efficiencies considered in our simulation refer to the higher calorific value. Finally, the efficiencies used in two large scale studies of Swiss energy consumption [27] [28], once converted to refer to the higher calorific value based on [29], are reproduced in Table V. Based on these observations, the default efficiencies of the main technologies were adapted in the third version to the values presented in Table III.

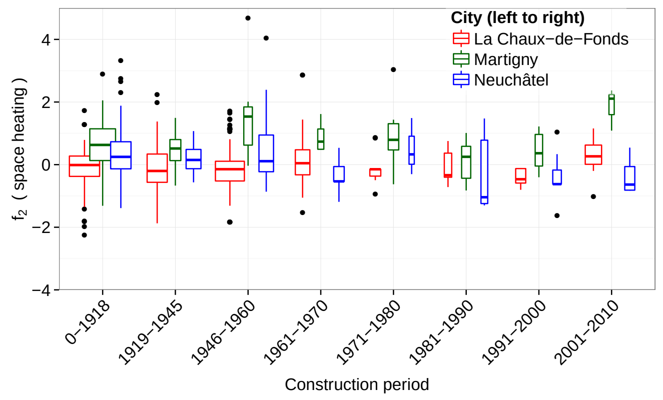


Figure 8. Discrepancy factor of the heating demand as a function of the construction period for the version 3 simulations. Four points outside the range of the graph were ignored.

C. *Version 3 simulations*

In addition to the ECS efficiencies modifications, the default properties of buildings were also adapted again, with the overall same hypotheses as before regarding pre-1960 constructions. For more recent constructions, the envelope's quality was lowered again, but most importantly the ventilation rate was substantially increased. We thus make the hypothesis that the usual ventilation habits clearly exceed the minimal recommended ventilation rate of 0.3 h^{-1}.

The simulation of all case studies with the third version of default values leads to the discrepancy factor for space heating showed in Figure 8. This third version intends to represent the most equilibrate hypotheses that can be made regarding the unknown parameters of our simulation, based on the available information and monitored data. A more refined calibration of the model for buildings built after 1960 would require a larger number of buildings to be relevant.

With this better calibrated space heating energy demand simulation, the f_2 factors for the DHW and electricity demands were also investigated. Both energy demands are estimated based on the building type; Figure 9 shows the f_2 factor per type for all services. The average electricity and DHW consumption of housing and administrative buildings was quite well estimated by the simulation based on norms, while other building types show, as expected, much less predictable trends.

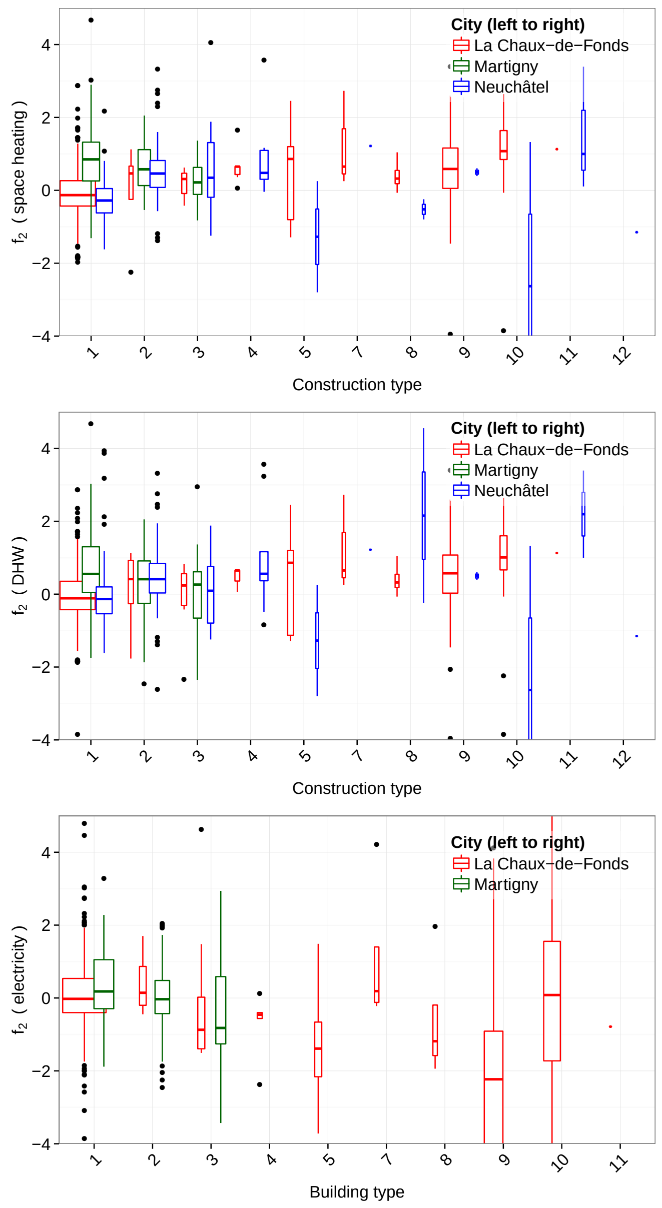


Figure 9. Discrepancy factor for the space heating, DHW and electricity services demands as a function of the building type in the version 3 simulations. Space heating and DHW show similar f_2 trends, as both are provided by the same ECS in most buildings. Allocation types: 1 apartment building, 2 individual home, 3 administrative, 4 schools, 5 sales, 6 catering, 7 meeting venues, 8 hospital, 9 industries, 10 warehouses, 11 sports installations, 12 indoor swimming pool.

VI. Discussion

Considering the case studies of CdF and Nch only, half of the housing and administrative buildings' f_2 value for space heating is comprised in the interval (-0.40, 0.43), meaning that the simulated heating demand of half the buildings was comprised between 76% and 132% of the monitored values. This can be considered as a good result for such a crude model, although the results for individual buildings cannot be trusted. The quality of the results for DHW is similar, while the interquartile range for electricity is (-0.30, 0.45). The simulation of other building types is much less reliable, and the results of the case study of Martigny remain uncertain.

The origin of the remaining discrepancies are numerous, but difficult to take into account. The space heating demand of individual homes is slightly overestimated when compared to apartment buildings, although the available data is not conclusive. Otherwise, the f_2 factor did not exhibit any significant correlation with the other available parameters that were tested (treated floor area, form factor, number of floors).

Among the inaccessible factors, the stochastic influence of occupants behaviour is known to be of high importance, practically limiting the precision of the results even with a very well calibrated physical model. However, numerous other sources of imprecision are known, in particular regarding space heating demand simulation:

- Many uncertainties regarding the correct attribution of monitored data remain. Visual representations of the f_2 factor help to spot likely errors, but more information is often needed to resolve them. For instance, adjacent buildings with high and opposed f_2 factor hints for a shared use of the energy consumption, but this often cannot be confirmed without on-site surveys.
- The existence of other ECS such as solar thermal panels and wood stoves is usually not documented and could not be assessed for this study. Aerial photography might prove to be a valuable source for the localisation of solar technologies, whereas the location of other technologies might remain very hard to assess without extensive surveys.
- The cadastral footprint of buildings used for the creation of the 3D model does not always represent the simulation relevant part of the building: some have been found to include adjacent garages, while buildings with a complex shape are simplified to the point where the 3D model and treated floor area estimation might not have any relevance at all. The use of the correct roof shape (instead of the simplified flat-roof model currently used) could also improve the simulation results' quality.
- The unknown refurbishment status of buildings is likely to account for an important part of the dispersion of the f_2 factor for all but the most recent buildings.
- Construction techniques are quite variable even for the same period, and further might depend on the region, although the difference between the three case studies simulated here with the same default values are not conclusive in this regard. The case study of Mrt, where a better thermal efficiency of buildings could be hypothesised based on the f_2 factor, is actually supposed to be a quite low energy efficiency neighbourhood.

Any attempt to further improve the calibration of the model without first addressing these uncertainties would not be pertinent. However, as the creation of models at this urban scale is likely to often suffer of the same limitations, it is interesting to document the precision of such models. The results obtained for buildings with monitored energy consumption also help to evaluate the reliability of the simulation on other buildings.

The simulated and adapted total space heating demands are shown in Figure 10, confirming for the case study of Mrt that either the space heating demand is overestimated, or the related consumption is underestimated. CdF and Nch total space heating demands are underestimated by 15.5% and 10.4%, although their average f_2 factors correspond respectively to a 5% underestimation and a 13% overestimation. Figure 11 shows that the highest space heating demands are indeed most frequently underestimated, and have an order of magnitude close to the difference between the total simulated and monitored values. Unlike regular statistical variations, the unpredictable demands of a few big energy consumers can thus have a strong impact on the overall results.

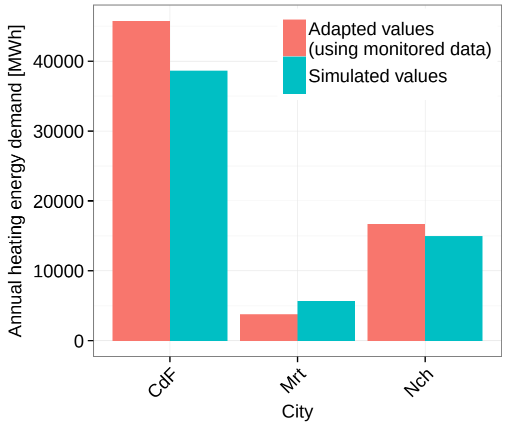


Figure 10. Annual space heating demand for housing and administrative buildings with monitored consumption.

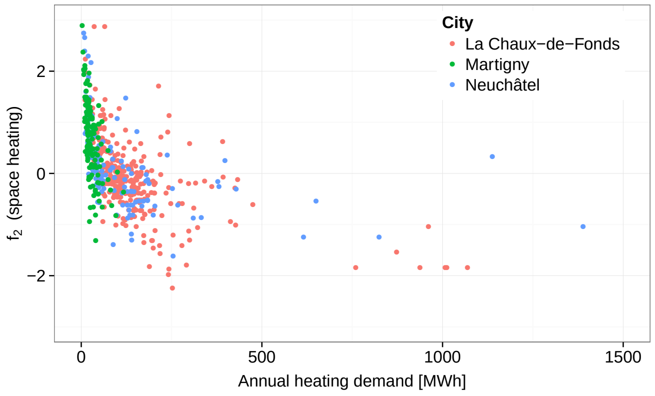


Figure 11. Correlation of the f_2 factor with the heating demand, as adapted based on monitored values. Four points with annual heating demand lower than 100 MWh and $f_2 > 3$ are outside the range of the graph.

Plotting the final energy use for heating versus the construction period of the buildings (Figure 12) does not reveals a clear decrease with time, except for the most recent buildings (built after 2000). All case studies present more or less the same trend, with only a marginal number of old buildings showing a clearly higher energy consumption for heating. The three partner cities have been promoting energy efficiency for some time and have all obtained labels in this domain [30]; nevertheless, Martigny's case study zone in particular is considered to have quite a low energy efficiency. On the other hand, the small number of recent buildings and possible errors in their modelling might have created a bias, which would require a broader study to be correctly explored.

VII. Conclusion and Future Work

The calibration and verification work presented in this paper improved and assessed the quality of the urban energy flow

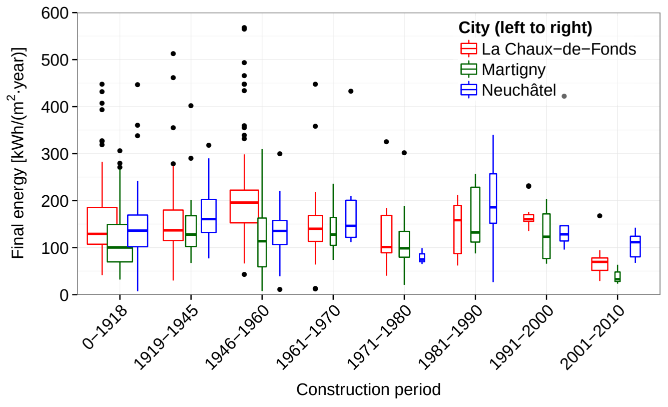


Figure 12. Final energy consumption for heating, per year and per square meter, for housing and administrative buildings with monitored consumption, as a function of the construction period.

simulation performed by the MEU platform. The platform intends to provide a flexible tool for the energy management at the scale of a neighbourhood, based on the limited available data sources and providing default values to supplement their low level of detail. The platform's combined management of simulation results and monitored data proved to be a powerful tool both to improve the model's quality and to perform verification studies.

The validity of the results at the scale of individual buildings was explored in detail, focusing on the housing and administrative buildings. The correct simulation of average buildings was demonstrated, while individual results remain, as expected, less reliable. The remaining discrepancies between simulated and monitored values can be attributed to the influence of the stochastic behaviour of occupants, but also to the limitations of the model regarding, among others, the attribution of monitored values, the unknown refurbishment status, the possible existence of other ECS, and the crude 3D model used for the simulation. Nevertheless, this study's results will already help improve the platform's reliability.

The aggregated results at the scale of a few hundred buildings shows that the total space heating demand is underestimated by 10% to 16% by the simulation for the two reliable case studies. This result highlights that a better match at the level of individual buildings does not necessarily yield a correct aggregated value, as some unpredictable high energy consumers can have an important impact. The possibility to simulate other buildings types with a satisfactory accuracy based on similarly low level-of-detail data remains to explore.

The research of realistic default values leads to new hypotheses regarding what can be considered as standard ventilation rates or as typical construction properties, but the lack of knowledge in this area makes the verification of those hypotheses difficult. A large scale but detailed assessment of existing buildings' physical properties would thus be a contribution of great value for the simulation of buildings' energy demand and for studies regarding the potential of energy-efficiency measures.

Finally, this study also evidenced interesting results regarding the energy use of our three case study areas. The most surprising observation is the low difference in energy consumption for space heating among buildings built before 2000. Note however that the individual renovation status of buildings is not known, and thus the role of recent refurbishments cannot be assessed. An analysis on a larger number of buildings built after 1960 would also be necessary to confirm the trends observed here.

ACKNOWLEDGMENT

The authors would like to thank the Swiss Federal Office of Energy (project number 102775) and all funders and partners of the MEU project, for providing most of the data used in this project, and for their help and availability to control the models and discuss the simulation results.

REFERENCES

[1] D. Perez, C. Vautey, and J. Kämpf, "Urban energy flow microsimulation in a heating dominated continental climate," in *Proceedings of SIMUL 2012, The Fourth International Conference on Advances in System Simulation*, 2012, pp. 18–23.

[2] L. Swan and V. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8, pp. 1819–1835, 2009.

[3] J. Keirstead, M. Jennings, and A. Sivakumar, "A review of urban energy system models: Approaches, challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3847–3866, 2012.

[4] L. Shorrock and J. Dunster, "The physically-based model BREHOMES and its use in deriving scenarios for the energy use and carbon dioxide emissions of the UK housing stock," *Energy Policy*, vol. 25, no. 12, pp. 1027–1037, 1997.

[5] J. Snäkin, "An engineering model for heating energy and emission assessment – The case of North Karelia, Finland," *Applied Energy*, vol. 67, no. 4, pp. 353–381, 2000.

[6] A. Egan, "Three case studies using building simulation to predict energy performance of australian office buildings," in *Proceedings of the eleventh international IBPSA conference, Glasgowleventh International IBPSA Conference*, 2009, pp. 896–903.

[7] Y. Shimoda, T. Fujii, T. Morikawa, and M. Mizuno, "Residential end-use energy simulation at city scale," *Building and environment*, vol. 39, no. 8, pp. 959–967, 2004.

[8] D. Robinson, F. Haldi, J. Kämpf, P. Leroux, D. Perez, A. Rasheed, and U. Wilke, "CitySim: Comprehensive micro-simulation of resource flows for sustainable urban planning," in *Proceedings of Building Simulation*, 2009.

[9] A. Strzalka, J. Bogdahn, V. Coors, and U. Eicker, "3D City modeling for urban scale heating energy demand forecasting," *HVAC&R Research*, vol. 17, no. 4, pp. 526–539, 2011.

[10] D. Perez and D. Robinson, "Urban energy flow modelling: A data-aware approach," *Digital Urban Modeling and Simulation*, pp. 200–220, 2012.

[11] EcoInvent, "EcoInvent Center, Swiss Center for Life Cycle Inventories," http://www.ecoinvent.ch, last checked : 03.12.2013, 2013.

[12] Société suisse des ingénieurs et des architectes, "Swiss norm 380/1 : L'énergie thermique dans le bâtiment," SIA Zurich, 2009.

[13] P.-A. Haldi and D. Favrat, "Methodological aspects of the definition of a 2kW society," *Energy*, vol. 31, no. 15, pp. 3159–3170, 2006.

[14] D. Robinson, A. Rasheed, J. H. Kämpf, M. Bruse, K. Axhausen, F. Flourentzou, M. Batty, F. Haldi, and D. Perez, *Computer Modelling for Sustainable Urban Design: Physical Principles, Methods and Applications*, D. Robinson, Ed. Routledge, 2011.

[15] S. Coccolo, J. H. Kaempf, and J.-L. Scartezzini, “Design in the desert. A Bioclimatic project with urban energy modelling,” in *Proceedings of Building Simulation 2013: 13th Conference of the International Building Performance Simulation Association*, France, 25-30 August 2013.

[16] A. Cifuentes Cuéllar, “Simulation urbaine d’un quartier de logement social à Bogotá, Colombie (Master thesis),” EPFL, Solar Energy and Building Physics Laboratory, 2012.

[17] T. Olofsson, S. Andersson, and J.-U. Sjögren, “Building energy parameter investigations based on multivariate analysis,” *Energy and Buildings*, vol. 41, no. 1, pp. 71–80, 2009.

[18] L. Perez-Lombard, J. Ortiz, and C. Pout, “A review on buildings energy consumption information,” *Energy and Buildings*, vol. 40, no. 3, pp. 394–398, 2008.

[19] D. G. Sanchez, B. Lacarrière, M. Musy, and B. Bourges, “Application of sensitivity analysis in building energy simulations: Combining first- and second-order elementary effects methods,” *Energy and Buildings*, 2012.

[20] F. d’Ambrosio Alfano, M. Dell’Isola, G. Ficco, and F. Tassini, “Experimental analysis of air tightness in mediterranean buildings using the fan pressurization method,” *Building and Environment*, vol. 53, pp. 16 – 25, 2012.

[21] W. R. Chan, W. W. Nazaroff, P. N. Price, M. D. Sohn, and A. J. Gadgil, “Analyzing a database of residential air leakage in the united states,” *Atmospheric Environment*, vol. 39, no. 19, pp. 3445–3455, 2005.

[22] A. Parekh and P. Eng, “Development of archetypes of building characteristics libraries for simplified energy use evaluation of houses,” in *Proceedings of the 9th International IBPSA conference*, 2005, pp. 921–928.

[23] Société suisse des ingénieurs et des architectes, “Swiss norm 2031 : Certificat énergétique des bâtiments,” SIA Zurich, 2009.

[24] SuisseEnergie, “Dimensionnement des chaudières à mazout et à gaz,” Swiss Federal Office of Energy, 2000.

[25] E. Jochem, G. Andersson, D. Favrat, H. Gutscher, K. Hungerbühler, R. von Rohr, D. Spreng, A. Wokaun, and M. Zimmermann, *Steps Towards a Sustainable Development: A White Book for R & D of Energy Efficient Technologies*. Novatlantis, 2004.

[26] Université catholique de Louvain (Belgique), Département de l’Énergie et du Bâtiment Durable., “Energie+ version 7,” http://www.energieplus-lesite.be/index.php?id=10396, last checked 27.03.2013, 2013.

[27] P. Hofer, “Der energieverbrauch der privaten haushalte, 1990-2035: Ergebnisse der szenarien i bis iv und der zugehörigen sensitivitäten bip hoch, preise hoch und klima wärmer,” Swiss Federal Office of Energy SFOE, Tech. Rep., 2006.

[28] H. Wallbaum, N. Heeren, M. Jakob, M. Gabathuler, N. Gross, and G. Martius, “Gebäudeparkmodell sia effizienzpfad energie dienstleistungs-und wohngebäude-vorstudie zum gebäudeparkmodell schweiz–grundlagen zur überarbeitung des sia effizienzpfades energie,” Swiss Federal Office of Energy SFOE, Tech. Rep., 2009.

[29] R. Frischknecht, M. Stucki, K. Flury, R. Itten, and M. Tuchschmid, “Primärenergiefaktoren von energiesystemen,” ESU-Services GmbH, fair consulting in sustainability, Uster, 2008.

[30] G. Cherix, M. Finger, M. Capezzali, H. B. Püttgen, A. Chapuis, and A. Nour, “Action and influence of the multiple decision levels over the whole energy chain,” in *Proceedings of the 5th Dubrovnik Conference on sustainable development of energy, water and environment systems*, Dubrovnik, 2009.

Design Space Exploration of Many-Core NoCs Based on Queueing-Theoretic Models

Erik Fischer, David Öhmann, Albrecht Fehske, and Gerhard P. Fettweis
Vodafone Chair Mobile Communications Systems
Technische Universität Dresden
Dresden, Germany
Email: {erik.fischer, david.oehmann, albrecht.fehske, fettweis}@ifn.et.tu-dresden.de

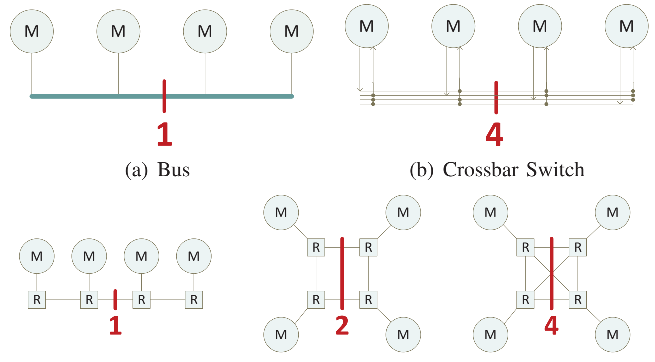
***Abstract*—The design of many-core system-on-chips confronts the developer with a more and more challenging task. Modern embedded applications have a continuously increasing requirement for highly parallelized and flexible heterogeneous processor structures. The interconnection problem becomes a crucial design decision with a growing number of parallel cores. Today, these decisions are usually solely based on the designer's experience. However, this will not be feasible anymore for future many-core systems with thousands of cores on a single chip. Automated guidance and tool support is essential to assist the design of network-on-chips, a common solution for the interconnection of modern system-on-chips. In this paper, we introduce a fast, flexible and accurate analytic model based on queueing theory to analyze the traffic in network-on-chips. The model requires only limited knowledge of the system and is therefore well-suited for the early phase of the design space exploration. It provides a high flexibility in terms of supported topology, routing scheme and traffic pattern, and enables to derive various performance metrics based on the steady-state distribution of the network routers. We evaluate the analytic model against cycle-accurate simulation and demonstrate its application based on some simple design examples, e.g., for buffer dimensioning, localizing bottlenecks, and benchmarking topologies. Several extension of the basic model are proposed to consider finite buffers, dynamic traffic, and to generalize the service time assumptions made for the network routers. This further increases the accuracy of the basic analytic model and expands its application area.**

***Keywords*-network-on chip; queueing theory; design space exploration; router model; transient behavior**

I. Introduction

In recent years, analytic models gain in importance to handle the growing complexity for designing and interconnecting multi-processor system-on-chips (MPSoC). In this paper, we present an extended work of the queueing model for network-on-chip (NoC) based MPSoC introduced in [1].

In embedded computing, todays applications show a common trend towards a continuously increasing computational effort and reliability. This is especially true in the area of multi-media and mobile communication. These requirements can only be fulfilled by massively exploiting parallelism. Emerging technologies like 3D chip stacking [2] promise a significant boost of the number of processor cores per mm^2. The technology allows the vertical stacking of multiple chips, e.g., by using through silicon vias, inductive or capacitive coupling or optical interfaces. It is expected that it will be



(a) Bus (b) Crossbar Switch

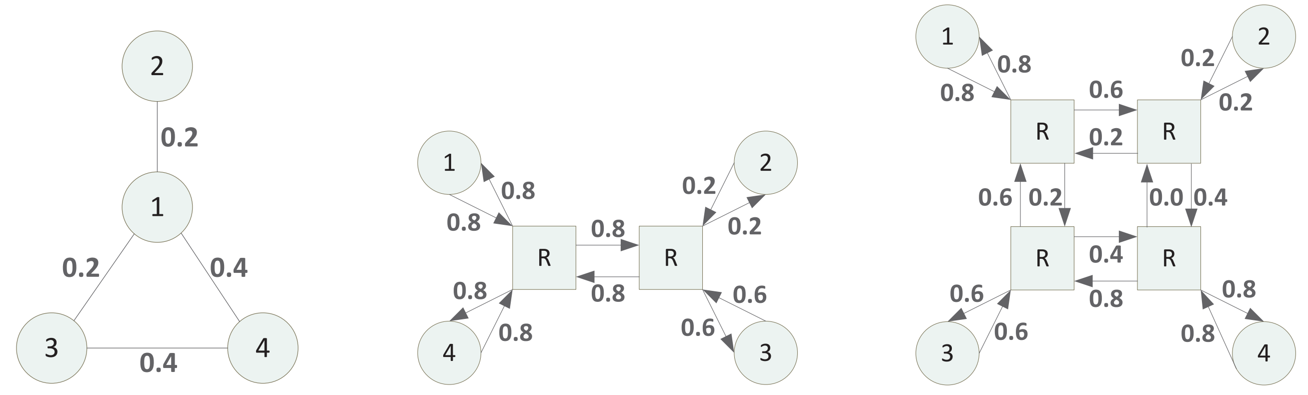
(c) NoC: chain (left), 2D-mesh (middle), fully connected (right)

Figure 1. Different alternatives for the interconnection of an MPSoC with 4 modules (M). (R=router)

possible soon to build stacks of hundreds of active layers, i.e., layers with processor elements or memories. By exploiting the third dimension and taking the ongoing technology scaling into account, it is expected that todays MPSoCs soon scale to many-core SoCs with thousands of processors on a single chip [3]. Already in 2015, we may have 1000 or more cores on a chip [4]. This trend becomes already obvious today in the GPU area where existing solutions provide up to 512 parallel cores [5].

Besides the increasing number of cores, we also recognize a trend towards more heterogeneity on-chip. Though heterogeneous multi-processor architectures require a more sophisticated controlling, they enable a better target-oriented computation. I.e., for every computational task a core can be selected which fits best the requirements of the task. Combined with a smart power management concept, this enables building high efficient MPSoCs that offer a high computational performance and low power consumption at the same time. A prototype of a heterogeneous MPSoC has been published in 2008 [6]. The Tomahawk chip consists of six fixed-point vector DSPs and two scalar floating-point DSPs. In addition, an LDPC decoder ASIP, a deblocking filter ASIP and an entropy decoder ASIC is provided. A central control unit (CoreManager) is responsible for the task scheduling and for transferring the data and program code between global and local memories.

If we consider such heterogeneous many-core architectures, the interconnection problem becomes a serious challenge.



(a) Application characterization graph with four modules.

(b) Solution with two modules per router that take module affinity into account.

(c) Solution with one modules per router to increase bisection width.

Figure 2. DSE example with two alternative topologies.

Classical interconnection architectures, such as busses or crossbar switches, cannot offer the necessary flexibility to support the different requirements of the heterogeneous processor cores. Additionally, conventional interconnects offer no good scaling with respect to throughput or area overhead. NoC evolved as a flexible and high-performance solution for the interconnection problem during the last decade [7][8]. NoC is a packet-switched on-chip network where packets are forwarded from a source to a destination via several intermediate router nodes. Each router consists of:

- a small buffer for the intermediate storage of the incoming packets at every input (and/or output),
- a crossbar switch for connecting an input with an output depending on the target address of a packet,
- and a control logic that realizes the routing and arbitration protocol of the NoC.

The routing protocol controls the route that a packet takes from a certain source node to a destination node. The arbitration controls the contention resolution. I.e., it decides which packet is forwarded at first, if two packets arrive simultaneously on different inputs and need to be forwarded to the same output. Routers can be connected in an arbitrary network topology. In addition, one or more processing nodes can be connected to a router. They are called modules. Their functionality is thereby transparent to the NoC, i.e., this could be a processor, memory or an external interface. The smallest transfer unit, to be transmitted over a NoC, is called the flit (flow control digit).

Figure 1 demonstrates the advantage of NoC over conventional solutions for the interconnection. In Figure 1(a), the interconnection of four modules by a bus is depicted. Therein, the red number represents the bisection width of this infrastructure. The bisection width is defined as the minimal number of wires that have to be cut to disassemble the network into two disjoint, equal-sized parts. The minimal cut represents the bottleneck of the network. Therefore, the bisection width can be used as a rough performance indicator for the network throughput. As it can be seen in Figure 1(a), the bus infrastructure is quite limited with respect to achievable throughput, since its bisection width is only 1. This is an issue, if we consider a large number of processors, since the bisection width does not scale with the number of connected modules. In contrast, it can be seen in Figure 1(b) that the crossbar switch offers a very high throughput. The bisection width scales with the number of modules (4 in this case). The drawback of the crossbar switch is that its required chip area grows quadratically with the number of connected modules. In addition, the delay of the switching logic grows linearly with the number of connected modules. Therefore, this interconnection type is not feasible for a large number of modules. NoC is a very flexible solution as depicted in Figure 1(c). Depending on the selected topology (chain, 2D-mesh, or fully connected), the throughput can be adapted according to the application requirements. In this example, the bisection width can be varied between 1 and 4. Moreover, the scalability of the network also depends on the selected topology and is thus adjustable. While the throughput of the chain topology does not scale with the number of connected modules, the fully connected NoC offers a scalability that is equivalent to that of the crossbar switch. The 2D-mesh is an intermediate solution. Finally, NoC allows us to make very flexible design decisions to tradeoff network throughput and latency against chip area, number of wires and achievable clock frequency of the resulting circuit.

However, finding an optimal NoC interconnect for many-core SoCs is a very challenging task, since many different design objectives and constraints have to be considered, like choosing routing and switching methods, selecting topology, application mapping, etc. [9]. This leads to a huge design space. In the following, we discuss a small example to motivate the challenge and the complexity of this process, the so called design space exploration (DSE). Again, we assume that the designed MPSoC shall consist of four modules. The MPSoC is intended for a specific application. The communication requirements between the modules are known in advance, as shown in Figure 2(a). The figure shows the application characterization graph (APCG) [9] for the small example. An edge in the APCG indicates that there is a communication requirement between the two modules. The annotated numbers represent the necessary communication amount. Therein, bidirectional traffic is assumed with the same communication amount in both direction. From the DSE point of view, the numbers could just be thought of some abstract values that represent the affinity between the modules. More concrete, the numbers could represent the average traffic amount (e.g., flits/cycle). The objective of this DSE example is to find a NoC topology that minimizes the average latency. Additionally, the maximum number of modules per router is constrained to two.

For the given problem, it might be a good heuristic to select two modules with a high affinity, i.e., with a high communication requirement, and assign them to the same router to minimize the path latency. According to Figure 2(a), the highest affinity is between modules 1 and 4, as well as, 3 and 4. Unfortunately, we are only allowed to assign two modules to one router according to our previously defined constraints. Thus, we just select modules 1 and 4. This results in the topology that is depicted in Figure 2(b). As it can be seen, the link between the two routers is quite congested (0.8 flits/cycle). Therefore, it might be a good idea to spend one router per module, instead, to increase the bisection width of the network and relax the congestion on the inter-router links. The resulting topology is shown in Figure 2(c). Though the congestion is reduced on the inter-router link, there is an additional router in the path from module 1 to module 4, which increases the latency of this path. The question is: Which of the two solutions is better and yields the lower average latency? This is hard to answer without making a deeper analysis of the proposed solutions. Thus, the designer might make the wrong decision here so that the resulting MPSoC will not offer the expected performance due to a communication bottleneck.

This small example should motivate why fast and accurate NoC models will be required that give an insight into the system and enable us to reduce the design space already in early design stages. Cycle-accurate simulation based approaches are too slow for this purpose. For the case of many-core SoCs there will be a large number of design alternatives. Moreover, the big number of routers increases the simulation time per run significantly. Simple high-level system models (e.g., only considering the propagation latency and ignoring queueing delays), on the other hand, are able to provide results in very short time. Due to the high abstraction, however, these models loose quite some accuracy. More detailed analytic models provide a good tradeoff between both approaches and are thus well suited for the NoC exploration of a many-core SoC.

Basically, there are two different approaches for analytic NoC models. Models based on the Network Calculus [10] are intended to analyze latency and throughput bounds. Therefore, worst-case assumptions are made for the traffic that the modules inject into the NoC as well as for the service times of the routers to handle a packet. This type of models is well suited to analyze NoCs with respect to quality of service. However, for the purpose of DSE it is necessary to have a more comprehensive view of the system. In general, it is not a good idea to make design decisions only based on the worst-case behavior of the system. A second approach utilizes probabilistic traffic models based on queueing theory [11]. This type of models is much better suited for the purpose of DSE, since it provides more insight into the system, allows to derive distributions (e.g., for the router latency), as well as mean values, and is also suited to yield some information about service guarantees.

A. Overview

In this paper, we propose an analytic NoC model based on queueing theory that provides a high degree of flexibility regarding topology, routing and traffic scheme. In contrast to existing models, it is not restricted to mean value analysis but provides information about the state distribution functions of the routers. It enables us to derive a variety of performance metrics, such as mean latency, buffer usage or overflow probability, and makes the model a very flexible tool for NoC performance analysis. Furthermore, we discuss several extensions of the basic NoC model. They give an even deeper insight into the system and provide more valuable information to the designer or DSE tool to make their design decisions.

- The basic NoC model is limited to the analysis of the system behavior in steady-state. Especially for highly dynamic applications or reconfigurable NoCs, knowledge about the transient behavior of the network is of great interest, cf. [12], [13]. Such an analysis improves the understanding of complex processes and can help to design parameters accurately. Therefore, we present an extension of the basic NoC model that enables us to analyze the system behavior in the time domain and in combination with time-varying rates.
- Making an infinite buffer assumption is a good approach for the basic NoC model. It keeps the computational complexity low and allows to do some analysis on networks in an early DSE design stage where the concrete knowledge of the buffer sizes is still not available. However, the consideration of finite buffers is a valuable extension which makes the model more accurate and allows to model congestion in the network. Moreover, the extension allows to extract new performance metrics, such as blocking probabilities or traffic acceptance rates.
- The assumption of exponentially distributed service times is a good starting point for the development of the basic NoC model. Again, this condition decreases the computational complexity and is a feasible assumption, if the concrete service time distributions are still unknown in an early DSE design stage. Nevertheless, quite accurate approximate solutions are available in literature for estimating the behavior of M/G/1 queueing systems (QS). This allows us to make this generalization without increasing computational complexity.

The remainder of this paper is structured as follows. In Section II, we discuss related work. The necessary mathematical fundamentals of queueing theory are provided in Section III. Section IV introduces the basic analytic NoC model. Starting with the system model and its assumptions (IV-A), the analytic model is discussed on network level (IV-B) and router level (IV-C). We evaluate the accuracy of the proposed approach against cycle-accurate simulation in Section V and provide some DSE application examples. Model extensions are proposed in Section VI for analyzing the transient network

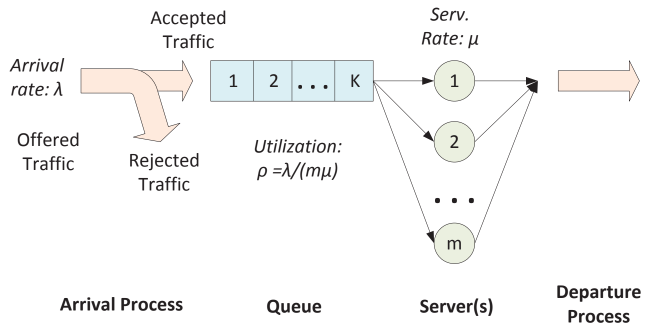


Figure 3. General idea of a queueing system.

behavior (VI-A), considering finite buffers at the router inputs and dealing with arbitrarily distributed service time processes (VI-B). Finally, Section VII concludes the work.

II. State of the art

Much effort has been spent for more than two decades for finding adequate traffic models for the analysis of off-chip and (later) on-chip networks. In 1990, Dally [14] developed analytic tools for investigating latency and throughput in networks, but restricting to k-ary n-cube topologies. Recent approaches focus on the mean value analysis of latency, throughput and energy consumption. Kiasari et al. presented an M/G/1 queueing model for wormhole switched two-dimensional (2D) torus NoC topologies, assuming deterministic routing [15]. This work was extended to G/G/1 queues in [16], which enables the analysis of bursty traffic. Another queueing-theoretic model focusing on buffer allocation was proposed by Hu et al. in [17]. A different approach was published in 2009 in [18], which introduces an empirical model to estimate contention delays for constant service time routers. Thereby, the hybrid router model takes into account Poisson input flows as well as output flows from preceding constant service time routers. Ogras et al. presented a fast and flexible analytic approach in 2010 [19] for the mean value performance analysis of virtual channel first-come first-serve (FCFS) input buffered routers for arbitrary topology and service time distribution. Other recent approaches for modeling on-chip networks [20] focus on the theory of the Network Calculus [10]. This theory provides a powerful tool for an estimation of performance bounds in NoCs, which is essential for giving statements about the realtime capabilities of a network in early design stages. However, for the exploration of the average network behavior, other methods, like stochastic models, are more expedient.

III. Fundamentals of Queueing Theory

For detailed studies of queueing theory in combination with network modeling we refer to [21], [22] and [11]. Subsequently, we give a short overview of the fundamentals. The basic understanding of queueing systems and Markov models is required in Section IV-C to follow the derivation of the router model.

A basic multi-server queueing system (QS) is depicted in Figure 3. A QS models the incoming stream of customers (here: flits) as a stochastic arrival process with a known distribution and mean arrival rate λ. The arrival stream is passed to a queue with a fixed number of K waiting slots. Note that K can also be zero or infinite. If no waiting slot is free upon arrival of the customer, the customer is rejected. Therefore, the accepted traffic that arrives at the queue is in general not equal to the offered traffic at the input of the QS. This is only the case, if an infinite queue is assumed. The customers in the queue are forwarded to m (parallel) servers in a certain order, which is defined by the service discipline of the QS (e.g., FIFO, LIFO). The service (i.e., service time) is again modeled by a stochastic process with a known distribution and mean arrival rate μ. The served customers of the m servers leave the QS as a combined departure process. Its mean rate is equal to that of the accepted traffic. A queueing system can mainly be characterized by four parameters:

- the type of arrival process (A),
- the type of service process (B),
- the number of servers (m),
- and the number of waiting slots (K).

An easy way to describe a QS is provided by the Kendall notation [23]: A/B/m/K. Examples for the arrival and service time process are: exponential/memoryless (M), constant/deterministic (D), Erlang-k-distributed (E_k), general (G). If the number of waiting slots is infinite, K is often omitted. Some examples are: M/M/1, M/D/1, G/G/1/K, M/G/m, etc.

The models presented in this work (Section IV-C) are based on the classical M/M/1 QS, which can be characterized by closed-form expressions. The simplicity of this queueing system arises from the Markov property (also called memoryless property) which is an inherent property of the exponentially distributed arrival and service process. It makes the system independent of past events, i.e., the next arrival/departure time is independent of the last arrival/departure time. As a consequence, the state of an M/M/1 system can fully be characterized by the probability distribution of finding a certain number of customers k in the system which is described by

$$\pi_k = (1-\rho)\rho^k.$$

Therein, ρ describes the average utilization of the queue, which depends on the relation between arrival rate and service rate and is defined as $\rho = \lambda/\mu$. An extended version of this probability distribution is applied in Section IV-C to derive the steady-state of NoC routers. Based on the distribution various performance indicators can be derived, like the average number of customers:

$$\overline{N} = E[\pi_k] = \sum_{k=0}^{\inf} k\pi_k = \frac{\rho}{1-\rho}.$$

A fundamental result of the queueing theory, known as Little's law, describes the relation between the average number of customers and the average time spent in the system (T):

$$\overline{N} = \lambda T.$$

This is a general result which is independent of the distribution of the arrival or service process. It enables the analytic

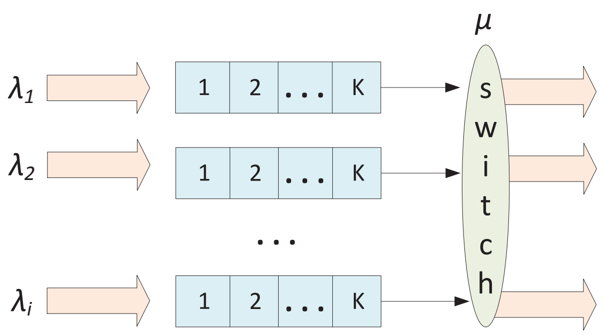


Figure 4. Modeling routers by multiple queueing systems.

computation of mean waiting times (i.e., latencies), which is an important performance measure for NoC, i.e.,

$$T = \frac{1/\mu}{1-\rho}.$$

Furthermore, it is possible to derive the tail probability based on the distribution of the number of customers. It represents the probability that the number of customers in the system exceeds a given capacity K:

$$P[k > K] = \sum_{k=K+1}^{\inf} \pi_k = \rho^{K+1}.$$

This is an interesting performance measure which can be employed for the task of buffer dimensioning (see Section V) and to provide certain service guarantees (e.g., maximum latencies).

The preceding expressions refer to a steady-state where transients have faded away and the system has reached stationarity. Our basic NoC model (Section IV) assumes steady-state which is sufficient for most analyses. However, for some purposes, e.g., analysis of startup behavior or time-varying traffic rates, it might be attractive to analyze transient characteristics as well. Therefore, we extend our basic model in Section VI-A. Unfortunately, analytical expressions for the transient behavior are often cumbersome or even hard to find [11]. In this case, numerical techniques can be applied (see Section VI-A).

Figure 4 depicts how the QS approach can be employed to model a single NoC router. Therein, every input is modeled by a separate QS. This is a natural mapping, since we assume indeed that every input has a separate buffer. The customer arrival stream is mapped to the stream of incoming flits with known mean arrival rates $\lambda_1, ..., \lambda_i$. Every input queue is served by a single server: the switch. Actually, all inputs are served by the same server. As mentioned before, an arbiter resolves contention cases. However, the contention resolution has a significant influence on the mean service rate for every input. For this purpose, a service time model has to be employed that decouples the input queues first under consideration of contention and arbitration policy. In [24], a service time estimation for round-robin arbitration has been proposed.

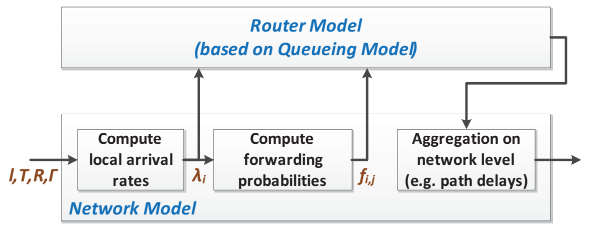

Router Model
(based on Queueing Model)
Compute local arrival rates
Compute forwarding probabilities
Aggregation on network level (e.g. path delays)
l,T,R,Γ
λi
fi,j
Network Model

Figure 5. The Hierarchical structure of the proposed analytic model.

IV. Basic NoC Model

Before we can start to introduce the analytic model, we need to define the system model and model assumptions first (Section IV-A). Some common restrictions are made to reduce the model complexity and keep it practicable. Still, we focus on preserve a high flexibility to serve a broad spectrum of applications. The basic NoC model is introduced in two steps, on network level (Section IV-B) and on router level (Section IV-C).

A. System Model

We assume the routers to be arranged in an arbitrary topology. An arbitrary number of cores is allowed to be connected to a single router. Due to the low buffer requirements, wormhole switching is the most favored switching technique for realizing best-effort services in on-chip networks today [9]. Therefore, we restrict our model to this technique. The routing protocol, on the other hand, shall not be restricted. Concerning the arbitration scheme, we restrict to the first-come first-serve method. Extensions to other arbitration schemes, like the popular round-robin, are feasible, as shown in [24]. Routers consist of an arbitrary number of buffered input ports and an arbitrary number of (unbuffered) output ports. In this section, we assume infinite buffer size.

Furthermore, we assume external packet arrivals from modules to possess Poisson characteristic [11], i.e., they have exponentially distributed inter-arrival times with known mean values. This assumption is often made to approximate real network traffic while reducing the model complexity at the same time. The router service times include processing delay for arbitration as well as forwarding delay for the packet and are assumed to be exponentially distributed. Furthermore, knowledge of the mean router service rate and router service latency is required. We assume it w.l.o.g. to be equal for all routers in the network. Finally, we imply a common clock for all routers.

To provide a flexible as well as a fast analytic model we propose to follow a hierarchical approach as depicted in Figure 5. We split the NoC model into an analysis on network level and on router level. By performing the analysis on router level and combining the results on network level, we thus reduce the complexity.

The network model receives multiple inputs that have to be specified by the user. The traffic scenario is described by the *traffic characterization matrix* $\boldsymbol{T}$ and the *external arrival rate vector* $\boldsymbol{l}$. The topology and interconnection are specified via the *connectivity matrix* $\boldsymbol{\Gamma}$. Finally, information about the

TABLE I: Model parameters and notation.

N_M	Number of modules
N_R	Number of router nodes
N_E	Number of edges
$\boldsymbol{T} = [t_{s,d}]$	Traffic characterization matrix (of size $N_M \times N_M$) with elements $t_{s,d}$ that specify the send probability from module s to module d
$\boldsymbol{l} = [l_s]$	External arrival rate vector (of size $N_M \times 1$) with elements l_s representing the arrival rate (packets/cycle) from source module s
$\boldsymbol{\Gamma} = [\gamma_{s,d}]$	Connectivity matrix (of size $(N_M + N_R) \times (N_M + N_R)$) with elements $\gamma_{s,d}$; $\gamma_{s,d} > 0$, if there is a directed connection from s to d; the value $\gamma_{s,d}$ represents the link ID for this connection ($\text{sgn}(\boldsymbol{\Gamma}) \equiv$topology matrix)
$\boldsymbol{R} = [r_{s,d,i}]$	Routing matrix (of size $N_M \times N_M \times N_E$) with elements $r_{s,d,i}$ defines the probability that link i is occupied for routing a packet from source module s to target module t ($\sum_{\forall i} r_{s,d,i} = 1$)
$\overline{x}$	Average router service time

applied routing scheme is provided via the *routing matrix* $\boldsymbol{R}$. An overview of the notation and a more detailed explanation is given in Table I.

Based on this information, the network model is able to compute local parameters for each router node individually, i.e., the inputs for the router model. The local parameters comprise the local arrival rates λ_i that is the accumulated arrival rate over all traffic flows that cross router input i. Furthermore, the forwarding probabilities $f_{i,j}$ are computed. $f_{i,j}$ defines the probability that a packet arriving at a router input i is forwarded to a router output j (please note that the indices i and j correspond to the unique identifier of the link that is connected to router input or output). The computation of the local arrival rates and forwarding probabilities is discussed in more detail in Section IV-B.

The local parameters can now be applied to a queueing model on router level. It is responsible for deriving the compound distribution for the number of packets in the input queues, which represent the router state. Consequently, the knowledge of the compound distribution enables a computation of key performance indicators, such as average buffer usage, overflow probabilities or mean queueing delays. The queueing model on router level is introduced in Section IV-C.

Finally, the performance metrics, computed on router level, have to be combined on network level, e.g., to derive path delays by summing up the queueing delays and the fix router propagation latencies.

B. Network Model

We can derive the vector of local arrival rates $\boldsymbol{\lambda}$, with elements λ_i ($1 \leq i \leq N_E$), by summing up all traffic flows that cross a specific link (and router input queue, respectively). Therein, N_E is the number of links in the network. The traffic characterization matrix $\boldsymbol{T}$ provides information about a pairwise traffic flow probability between each module s and d. By weighting $\boldsymbol{T}$ with the external arrival rates $\boldsymbol{l}$, we get the traffic intensities (in packets/cycle) for each pair of modules. Finally, we multiply the traffic intensities with the probability that the flow will pass link i (given by routing matrix $\boldsymbol{R}$) and sum up the fractions of the contributing traffic flows:

$$\lambda_i = \sum_{s=1}^{N_M} \sum_{d=1}^{N_M} l_s \cdot t_{s,d} \cdot r_{s,d,i}, 1 \leq i \leq N_E. \tag{1}$$

The notation is given in Table I. By applying the definition of the Frobenius inner product [25], we can rewrite (1) as matrix equation as follows:

$$\lambda_i = \text{tr}\left((\boldsymbol{L} \cdot \boldsymbol{T})^{\text{T}} \boldsymbol{R_i}\right). \tag{2}$$

In (2), tr(•) represents the trace of the matrix, $\boldsymbol{L}$ is the $N_M \times N_M$ diagonal matrix representation of vector $\boldsymbol{l}$:

$$\boldsymbol{L} := \text{diag}(\boldsymbol{l}),$$

and $\boldsymbol{R_i}$ the corresponding submatrix of $\boldsymbol{R}$ that consists of all elements $r_{s,d,i}$ with $1 \leq s, d \leq N_M$. We can select the set of local arrival rates Λ^r for a single router node r by exploiting the knowledge of the topology that is contained in the connectivity matrix $\boldsymbol{\Gamma}$. I.e., we collect all λ_i where i is the ID of an input edge of router r:

$$\Lambda^r := \left\{\lambda_{\gamma_{s,r}} \mid \gamma_{s,r} \neq 0; s \in \{1, \ldots, N_M + N_R\}\right\}. \tag{3}$$

We continue to compute the forwarding probability matrix $\boldsymbol{F}$. The matrix element $f_{i,j}$ ($1 \leq i, j \leq N_E$) can be defined as traffic intensity between router input i and router output j normalized to the total arrival rate at input i, i.e., λ_i:

$$f_{i,j} := \frac{\sum_{s=1}^{N_M} \sum_{d=1}^{N_M} l_s \cdot t_{s,d} \cdot r_{s,d,i} \cdot r_{s,d,j} \cdot \delta_{i,j}}{\lambda_i}, 1 \leq i, j \leq N_E. \tag{4}$$

We call the term $\delta_{i,j}$ the *link selector matrix*. It ensures that there is only a forwarding probability $f_{i,j} > 0$, if (i, j) represents an input/output link pair of the same router:

$$\delta_{i,j} := \begin{cases} 1, & if\ \exists s, r, d\ with\ \gamma_{s,r} = i\ \wedge \gamma_{r,d} = j \\ 0, & otherwise \end{cases}.$$

Therein, $\gamma_{s,r}$ and $\gamma_{r,d}$ are corresponding elements of the connectivity matrix $\boldsymbol{\Gamma}$. Equation (4) can be rewritten in matrix form:

$$f_{i,j} := \frac{\delta_{i,j}}{\lambda_i} \cdot \text{tr}\left((\boldsymbol{L} \cdot \boldsymbol{T})^{\text{T}} \left(\boldsymbol{R_i} \circ \boldsymbol{R_j}\right)\right), \tag{5}$$

where $\circ$ represents the entry-wise multiplication (i.e., the Hadamard product) of two matrices. Finally, we also restrict the set of forwarding probabilities F^r to a single router node r, similar to the approach in (3), and come to (6):

$$F^r := \left\{f_{\gamma_{s,r},\gamma_{r,d}} \mid \gamma_{s,r}, \gamma_{r,d} \neq 0; s, r \in \{1, \ldots, N_M + N_R\}\right\}. \tag{6}$$

C. Router Model

Based on the assumptions that we made in Section IV-A, an M/M/1 queueing system [11] with exponential interarrival and service times will be appropriate to model the router behavior. However, in reality, the traffic situation within a router looks more complicated, as the example in Figure 6a) shows.

Therein, we find splitting and merging of traffic flows that contend with other input queues for multiple output ports. Furthermore, each input has different probabilities of being

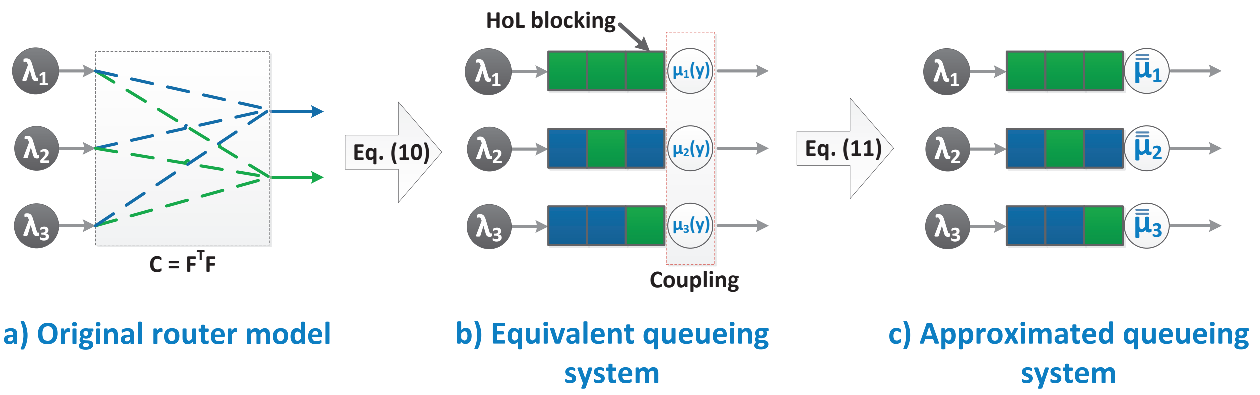


Figure 6. The original router model is transformed to an equivalent queueing model where the service rates of the input queues are mutually coupled. As a second step, an approximation by state aggregation is applied to decouple the queues.

forwarded to a specific output. To be able to represent the router system by a queueing model, we propose using a simplified equivalent system, as depicted in Figure 6b). The idea is to include the contention delays into the service times and thereby receiving port specific service times (or service rates, respectively). In fact, if a packet in front of a (FIFO) queue is blocked due to a contending queue, this is nothing else than a delayed service. Therefore, it is reasonable to consider the contention delay as a service time increase (or service rate decrease). Consequently, we come to a reduced equivalent system that consists of one queue per input, each with an individual server with a service rate $\mu_i(y)$, as shown in Figure 6b). Therein, the service rates depend on the current router state y, i.e., contention situation. The service rates decrease according to the degree of contention in the current router state. We recognize that the service of contending queues is still coupled.

Due to the memoryless property of the exponentially distributed arrival and service processes, the state of the equivalent router system can now solely be defined by the number of flits contained in the input queues. If we represent the state by a vector where each element represents the fill level of a single input queue, we can model the system by means of a multidimensional Markov chain. This is illustrated in Figure 7 for the case of a router with two inputs (please ignore the depicted macro states for now). Therein, the transition rates are defined by the arrival rate λ_i and service rate μ_i for each input independently. Let $\boldsymbol{x}$ be the current state vector of the router. Then, a transition from state $\boldsymbol{x} \rightarrow \boldsymbol{x} + e_i$ (where e_i is the i'th unit vector, i.e., a vector of all zeros except element i, which is equal to one) has an intensity of λ_i. On the other hand, a transition $\boldsymbol{x} \rightarrow \boldsymbol{x} - e_i$ has an intensity of μ_i. The boundaries of the Markov chain are an exception to that rule (first column/row in Figure 7). There, we find a different contention situation. In the case of two inputs, we have no contention caused by the second input anymore. Therefore, the transition rates for $\boldsymbol{x} \rightarrow \boldsymbol{x} - e_i$ change to μ, i.e., the basic router service rate without contention delay.

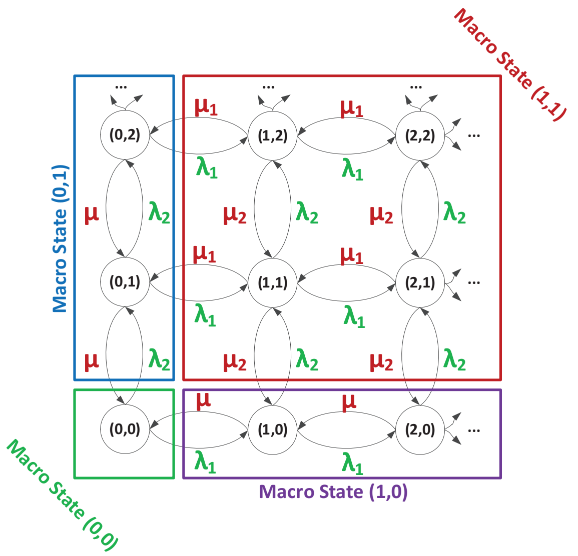


Figure 7. Example of a two-dimensional Markov model for a router with two inputs and the decomposition into reversible sub-chains.

For solving the Markov chain, we still need to know the service rates μ_i that include the contention delay to be able to define the transition rates. For this purpose, we apply an idea that was proposed in [19] to determine the mean waiting time for a similar input buffered router model assuming an FCFS arbiter. We modify this approach to find an estimation for the mean service time, i.e., the waiting time of the flit in front of the queue. Similar to [19], we first compute the pairwise contention probability $c_{i,j}$ for all inputs pairs (i, j) of a router with P inputs based on the forwarding probabilities $\boldsymbol{F}$ that can be derived according to (5):

$$c_{i,j} = \sum_{k=1}^{P} f_{i,k} f_{j,k}, i \neq j, 1 \leq i, j \leq P. \tag{7}$$

From (7), an equivalent matrix equation can be derived

$$\boldsymbol{C} = \boldsymbol{F} \cdot \boldsymbol{F}^{\mathrm{T}}. \tag{8}$$

Note that the main diagonal of the contention probability matrix $\boldsymbol{C}$ in (8) is set to "1" which makes the following computation more convenient. Based on the contention probabilities, we can derive an expression to estimate the mean service times

$\overline{x}_i(\boldsymbol{y})$ under contention:

$$\overline{x}_i(\boldsymbol{y}) := \overline{x} + \overline{x} \sum_{j=1, j\neq i}^{P} c_{i,j} y_j, 1 \leq i \leq P. \quad (9)$$

The first summand $\overline{x}$ of (9) represents the mean router service time for the packet in front of queue i. The second summand considers the contention delay. Therein, the vector $\boldsymbol{y}$ represents the instantaneous fill state of each input queue, i.e., $y_i = 0$, if input queue i is empty and does not contribute to the contention delay and $y_i = 1$ otherwise. We call $\boldsymbol{y}$ the *router macro state* in the following and can directly derive it from the router state $\boldsymbol{x}$:

$$y_i = \begin{cases} 0, & if\ x_i = 0 \\ 1, & if\ x_i > 0 \end{cases},$$

or rather informally: $\boldsymbol{y} = \mathrm{sgn}(\boldsymbol{x})$.

We can still condense (9) somewhat by exploiting the convenient definition of contention probability matrix $\boldsymbol{C}$ and provide a short form matrix equation for the mean service rates $\mu_i(\boldsymbol{y})$ (i.e., the inverse of the mean service times):

$$\mu_i(\boldsymbol{y}) := \left[\frac{1}{\mu} \boldsymbol{C}_i{}^{\mathrm{T}} \boldsymbol{y}\right]^{-1}, 1 \leq i \leq P. \quad (10)$$

With the definition for the mean service rates $\mu_i(\boldsymbol{y})$ in (10) we have now all necessary inputs to solve the Markov chain in order to obtain the steady-state probability distribution. However, in trying to do so, we are confronted with another challenge. If we apply the Kolmogorov criterion for reversibility of Markov chains, we soon realize that it does not hold for some cases in the peripheral region of our Markov chain. Accordingly, the chain is not time reversible; see Figure 7 and examine the following state transitions: $(0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (0,1) \rightarrow (0,0)$, and the corresponding return path. We notice that the product of the transition rates is not equal for both directions, and thus, it does not fulfill the Kolmogorov criterion [26]:

$$\lambda_1 \cdot \lambda_2 \cdot \mu_1 \cdot \mu \neq \lambda_2 \cdot \lambda_1 \cdot \mu_2 \cdot \mu.$$

Consequently, we are not allowed to apply local balance equations to solve the chain. Unfortunately, we are not able to find a closed-form solution for the infinite Markov chain solely based on the global balance equations. Fehske and Fettweis [27] recently encountered exactly the same problem when trying to solve an equivalent Markov chain. They proposed an approximation to find a solution for the stationary distribution. The approach is based on the concept of *aggregation of variables* that is well known by economics for quite some years [28]. The proposed algorithm consists of the following steps.

We start decomposing our Markov chain into reversible sub-chains. This is done by collecting all states $\boldsymbol{x}$ that belong to the same macro state (or aggregate state) $\boldsymbol{y} = \mathrm{sgn}(\boldsymbol{x})$ in a common set $\mathcal{S}(\boldsymbol{y})$:

$$\mathcal{S}(\boldsymbol{y}) := \left\{\boldsymbol{x} \in \mathbb{N}_0^P \mid \mathrm{sgn}(\boldsymbol{x}) = \boldsymbol{y}\right\}.$$

The idea behind the definition is that all states are collected in the macro state where we find a similar contention situation. If we consider a contending queue, it does not matter how many packets it contains, only if it contains at least one packet or not. Consequently, the mean service rates are homogeneous within each macro state. Thus, service rates are decoupled by this aggregation approach, as Figure 6 c) shows. An example for the Markov chain decomposition for the case of two input ports is provided in Figure 7. Therein, we decompose the two-dimensional Markov chain into four macro states. Macro state $(0,0)$ contains all states where both input queues are empty (which is only a single router state $(0,0)$). Macro states $(1,0)$ and $(0,1)$ collect the states where only one of the two queues is empty. Hence, we have no contention within these two macro states. Macro state $(1,1)$ represents all router states where both queues are not empty. In this two-dimensional example, this is the only macro state where contention occurs.

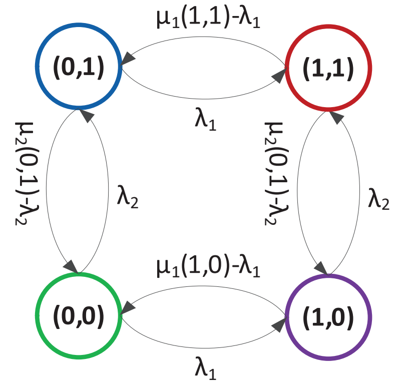


Figure 8. Example: Markov chain on macro state level assuming a router with two inputs.

Since the transition rates are homogeneous within each macro state, the sub-chains are reversible and can be solved. This leads to a product form solution for the stationary probability distribution of the number of customers (i.e., packets) $\widetilde{\pi}$ in an M/M/1 queueing system that is well known from classical queueing theory [11][27]:

$$\widetilde{\pi}(\boldsymbol{x}) = \begin{cases} \prod_{i\in N_1(\boldsymbol{y})} (1-\rho_i(\boldsymbol{y}))\rho_i^{x_i-1}(\boldsymbol{y})\sigma(\boldsymbol{y}), & \text{for } \boldsymbol{y} \neq \boldsymbol{0} \\ \sigma(0), & \text{for } \boldsymbol{y} = \boldsymbol{0} \end{cases} \quad (11)$$

with utilization $\rho_i(\boldsymbol{y})$ of input queue i defined as

$$\rho_i(\boldsymbol{y}) := \frac{\lambda_i}{\mu_i(\boldsymbol{y})}.$$

The terms $\sigma(\boldsymbol{y})$ denote the probabilities of finding the system in macro state $\boldsymbol{y}$ (i.e., one of the states in $\mathcal{S}(\boldsymbol{y})$). Note that (11) only yields an estimate for the solution of the stationary probability distribution. This is because we omit the transitions between the macro states at this consideration. Also, note that (11) is conditioned on the probabilities of the corresponding macro state $\sigma(\boldsymbol{y})$ to ensure that $\sum_{\boldsymbol{x}} \widetilde{\pi}(\boldsymbol{x}) = 1$.

So far, we have no knowledge about the macro state probabilities $\sigma(\boldsymbol{y})$. We can compute $\sigma(\boldsymbol{y})$ by solving the (now finite) Markov chain on macro state level. Figure 8 shows a solution for the transition rate $p(\boldsymbol{y}, \boldsymbol{y}')$ from macro state $\boldsymbol{y}$ to macro

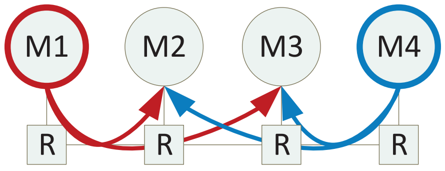


Figure 9. Topology and traffic pattern of first simple test scenario (M=module, R=router).

state $\boldsymbol{y}'$, as provided by [27]:

$$p(\boldsymbol{y},\boldsymbol{y}') = \begin{cases} \lambda_i, & \text{for } \boldsymbol{y}' = \boldsymbol{y} + e_i \\ \mu_i(\boldsymbol{y}) - \lambda_i, & \text{for } \boldsymbol{y}' = \boldsymbol{y} - e_i \\ 0, & \text{else} \end{cases}, \tag{12}$$

where e_i again represents the unit vector for dimension i. Based on (12), we can now define the transition probability matrix $P = \left[p_{ij}\right]$ with $p_{ij} := p\left(y_i, y_j\right)$. With the definition of

$$p_{ii} := -\sum_{j=1}^{2^P} p_{ij}$$

we normalize the row sum to 0.

Finally, we can follow the usual approach and solve the equation system for the vector of macro state probabilities σ based on the transition probability matrix P:

$$\sigma P = 0,$$

under the side condition $\sum_{\boldsymbol{y}} \sigma(\boldsymbol{y}) = 1$.

Based on (11), we can now compute the approximates for the state probabilities $\widetilde{\pi}(\boldsymbol{x})$. We can derive several key performance indicators, such as the mean number of packets in the queue $\mathbb{E}[x_i]$:

$$\mathbb{E}[x_i] \approx \sum_{\boldsymbol{x}} \widetilde{\pi}(\boldsymbol{x}) x_i = \sum_{\boldsymbol{y}} \frac{\rho_i(\boldsymbol{y})}{1-\rho_i(\boldsymbol{y})} \sigma(\boldsymbol{y}), \tag{13}$$

or the mean queueing delay W_i for input queue i by applying Little's law [11]:

$$W_i = \frac{\mathbb{E}[x_i]}{\lambda_i}.$$

V. Numerical evaluation

We show the accuracy of the proposed NoC model by comparing it against cycle-accurate NoC simulation. Due to the similar system model assumptions we decided to compare our approach against the model proposed in [19] as well as the NoC simulation tool that has been used therein [29].

We assumed following common simulation parameters:

- deterministic, dimension-ordered XY-routing,
- flit traffic, i.e., packet size = 1,
- input buffered routers with FCFS arbiter and service rates of $\mu = 0.5$,
- large buffer size (256 flits) to approximate the infinite buffer model, and
- simulation run time of 10^5 cycles with a warm-up period of 10^4 cycles.

We investigate the following two topology/traffic scenarios under different load conditions (defined by number of injected packets/cycle) and compare the average packet transmission latency in the network.

A. Introductory Example

First, we choose a very simple scenario to investigate the model behavior under a clear contention situation. Therefore, we consider a simple chain of four routers, as depicted in Figure 9, where a single module is connected to each router. The modules 1 and 4 are sending their packets to modules 2 and 3 with equal probability. Modules 2 and 3 do not send any packets. Hence, we find at router 2 and 3 a contention situation with the following forwarding probability matrix $\boldsymbol{F}$:

$$\boldsymbol{F} = \begin{pmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 1 & 0 & 0 \end{pmatrix}.$$

The result under different load conditions is shown in Figure 10. We find that the latency estimation for our proposed approach (red curve with + marker) follows very well the cycle-accurate simulation results (black curve with point marker) under a low and medium load condition. However, it significantly underestimates the network saturation limit where latency tends to infinity (0.66 packets/cycle in our model compared to 0.8 packets/cycle in the cycle-accurate simulation). The reference mean value model from [19] (blue curve with circle marker) shows a slight overestimation of the latencies under mid load conditions but estimates the network saturation point quite well.

The reason for the poor estimation of the network saturation point of our model is the applied aggregation approach for approximating the solution of a Markov chain. Therein, the stability of the overall solution is determined by the stability of the "worst-case" aggregate, i.e., the aggregate with the highest contention. If the solution for the "worst-case" aggregate tends to infinity the overall solution tends to infinity as well. To avoid this behavior, we propose to determine an average service time

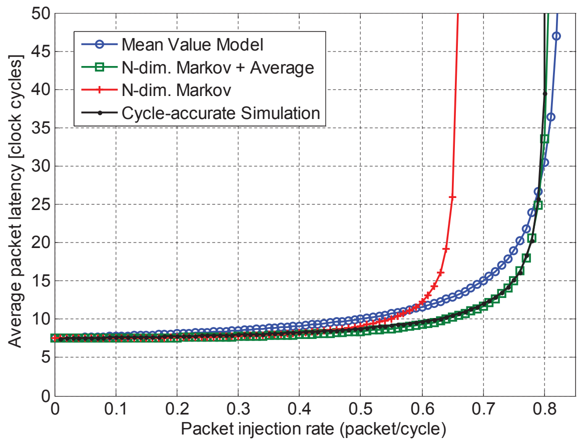


Figure 10. Performance results for 4x1 chain analyzing the average packet latency in comparison to cycle-accurate simulation.

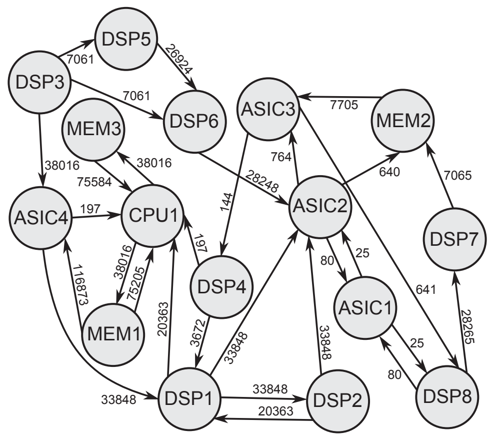


Figure 11. Traffic pattern of 4x4 2D-mesh test scenario with application-specific traffic [30] [19].

$\bar{\bar{x}}_i$ over all macro states for every router input. This is done by computing the expectation of the mean service times $\bar{x}_i(\boldsymbol{y})$ over all macro states based on the known macro state probabilities $\sigma(\boldsymbol{y})$:

$$\bar{\bar{x}}_i = \sum_{\boldsymbol{y}\in\{0,1\}^P} \bar{x}_i(\boldsymbol{y})\sigma(\boldsymbol{y})y_i. \tag{14}$$

Therein, y_i constrains the expectation to those macro states where queue i is not empty. We compute the average waiting time W_i for input queue i based on (14):

$$W_i = \frac{\bar{\bar{x}}_i}{1-\lambda_i\bar{x}_i}.$$

The result of the refined approach is also depicted in Figure 10 (green curve with square marker). It shows a very good match compared to the cycle-accurate simulation. The latencies under low/mid load conditions, as well as the network saturation point, are estimated very accurately by this approach. The average estimation error is less than 3%.

B. Multimedia application scenario

In addition, we choose a 4x4 2D-mesh topology using a more diverse traffic pattern of the generic multimedia application from [30]. The traffic scenario is depicted in Figure 11 while the topology and the core mapping is shown in Figure 12. We target to compare the estimation quality of the average latencies under more complex contention situations. The results are plotted in Figure 13 and confirm the accurate results of the first scenario. Again, the average estimation error is around 3% (9% for the reference model). However, we still notice a slight underestimation of the network saturation limit of about 2.5% for that case. The reference mean value model shows a better accuracy in this region.

Note that the presented results only serve as proof of concept. However, they easily scale to larger networks. The relative accuracy of the latency estimation is expected to stay in the same range under similar contention situations, independent of the NoC size because the analysis of the queueing delay is done on router level and only accumulated on network level.

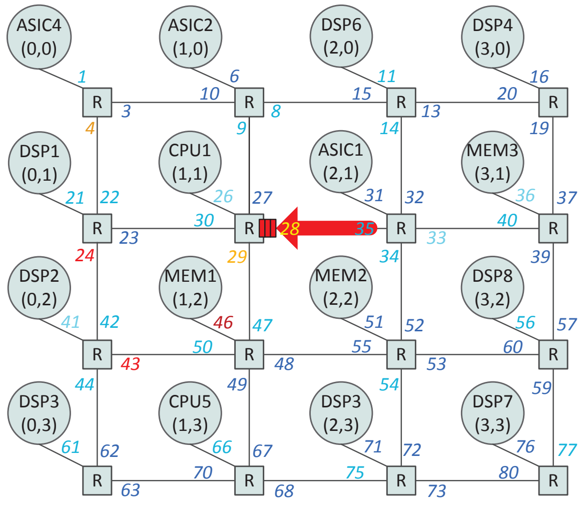


Figure 12. Core mapping based on 4x4 2D-mesh topology (R=router) [19]. The annotated numbers represent the index of the associated input buffer. The color denotes the tail probabilities ($P[x \geq 1]$) at the corresponding input buffers (red=high, yellow=medium, blue=low) which reveals the bottlenecks in the 4x4 2D-mesh.

C. Buffer dimensioning based on tail probability estimation

One advantage of the presented NoC traffic model is its flexibility to derive arbitrary performance metrics based on the distribution of the number of customers from (11). This is demonstrated in the following. An important step of the design space exploration for NoC based MPSoC is the so called buffer dimensioning. Therein, the necessary buffer capacity K is estimated for every router input for a given, topology, application (i.e., traffic pattern) and routing scheme. Memory consumes a lot of chip area and is therefore a crucial factor to reduce chip production cost. Hence, a careful investigation and optimization of the router memories is recommended. We employ the tail probability P_{tail} to invest the necessary buffer amount. This measure indicates the probability that a certain

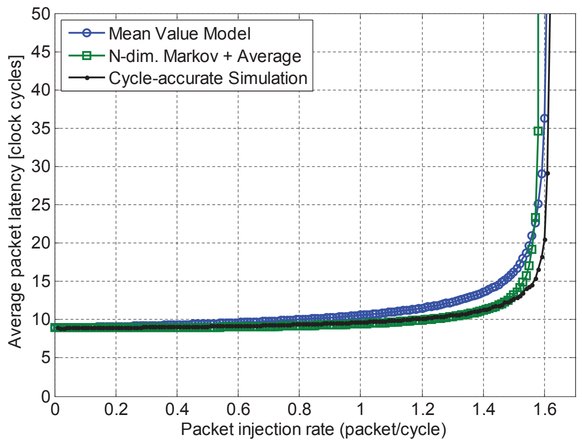


Figure 13. Performance results for 4x4 2D-mesh with generic multimedia application traffic analyzing the average packet latency in comparison to cycle-accurate simulation.

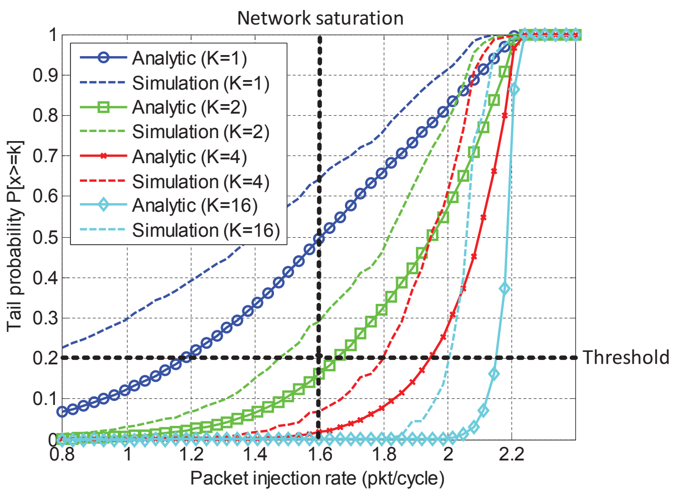


Figure 14. Analysis of the tail probability for a single router input buffer in a 4x4 2D-mesh with generic multimedia application traffic in comparison to cycle-accurate simulation.

buffer fill level is exceeded and is thus well suited for this purpose. It can easily be derived from the distribution of the number of customers $\widetilde{\pi}$ of the presented infinite buffer model.

$$P_{tail} = P[x_i \geq K] = 1 - \sum_{\boldsymbol{x}, x_i < K} \widetilde{\pi}(\boldsymbol{x}) \tag{15}$$

Equation (15) computes the tail probability for router input queue i. On this basis, we can now examine the buffer requirements focusing on a single router input of the presented 4x4 2D-mesh scenario. The location of the selected buffer (index 28) is illustrated in Figure 12. We investigate the tail probability for different injection rates (in range 0.8 to 2.4 packets/cycle) and different K (1, 2, 4, and 16). The results are again validated by comparison with cycle-accurate NoC simulations, as shown in Figure 14.

The figure yields much information that can help a designer to optimize the buffer according to the expected traffic volume. First, we define a threshold that is used to decide whether it is worth to investigate additional memory or not. We first consider the curves obtained from the analytical model. We take a tail probability of 0.2 as our threshold. This means that for a given K and traffic load, it is recommended to increase K, if the buffer is completely filled in at least 20% of the time. In Figure 14, we can see that under low and medium traffic load the threshold is not exceeded; even for a very small buffer size (K=1). At an injection rate of 1.2 packets/cycle, the curve "Analytic (K=1)" reaches the threshold and thus it is recommended to use a buffer size of 2 for this traffic load. This is sufficient up to an injection rate of 1.7 packets/cycle, where the green curve (Analytic K=2) exceeds the defined threshold. For higher injection rates it is recommended to use a buffer size of K=4. Only within the small region of 2 to 2.2 packets/cycle, it would make sense to use an even bigger buffer (K=16). For higher injection rates, the incoming traffic cannot be served anymore and the buffer is completely filled, independent of the buffer size. From our previous analysis of Figure 13, we know that the overall network saturation is already reached at an injection rate of about 1.6 packets/cycle. Taking this additional information into account, there is no need for over-provisioning the buffer by considering injection rates beyond network saturation. Finally, we can conclude from Figure 14 that a buffer size of only 2 is already sufficient to deal with all sensible traffic loads (up to 1.6 packets/cycle) for the given application scenario. Furthermore, Figure 14 depicts the results from the cycle-accurate simulation as reference. We find that the analytic model is able to represent the general behavior of the simulation quite well. However, we also see that it generally underestimates the tail probability due to the simplified assumptions of the arrival and service time distributions (M/M/1), as well as, the additional inaccuracy caused by the aggregation approach. Nevertheless, the influence on the design decisions is insignificant so that the proposed approach is well suited for the early DSE phases.

Up to now, we focused on the investigation of a single input buffer under different injections rates. If we fix the injection rate to a value close to the network saturation (1.6 packets/cycles), we are able to analyze the buffer requirements under worst-case conditions for the whole NoC at once. The results are presented in Figure 15 and provide the NoC designer an easily comprehensible and intuitive tool for the buffer dimensioning.

The figure illustrates the tail probability as color-coded blocks for all buffers of the network (x-axis) and different buffer sizes K (y-axis). Thereby, a blue block corresponds to a low P_{tail}, i.e., a quite relaxed traffic situation. A red block represents a high P_{tail}, and reveals potential bottlenecks. For these cases, it is suggested to increase the buffer size until reaching the green or blue region to optimize the traffic flow and avoid network congestion. We observe in Figure 15 that a quite low buffer size (K=1 or K=2) is sufficient for most buffers. Only a few buffers require some additional memory. E.g., a buffer size of 4 would be reasonable for input buffers 28 and 29. The bottleneck in this network scenario is clearly buffer 46 which is connected to the output of "MEM1". This is accordant to the traffic scenario in Figure 11, where MEM1 has a very high traffic load at its output. The diversity of the traffic load is caused by the application specific traffic pattern, which is clearly represented in Figure 15. The scenario demonstrates the advantage of a careful network analysis and buffer dimensioning quite well. Individual buffer sizing can save a lot of memory while ensuring a high performance at the same time.

Annotating the network topology with the corresponding color-coded tail probabilities (for K=1) at the input identifiers yields a clear picture concerning the localization of the bottlenecks. This is illustrated in Figure 12. We find the highest congestion around the modules "MEM1", "CPU1", "DSP1", and "DSP2". Relating to the traffic pattern in Figure 11, we can verify that these components are indeed highly interconnected, communication intensive, centric nodes.

As mentioned before, we observe a big diversity of the buffer loads due to the application scenario. Assuming a uniform traffic scenario (i.e., each module communicating with every other module with the same probability), we expect a more

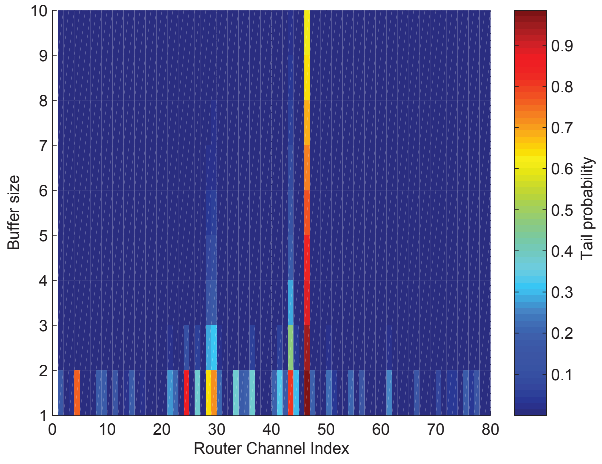


Figure 15. Buffer dimensioning for 4x4 2D-mesh with generic multimedia application traffic at fix injection rate (1.6 packets/cycle).

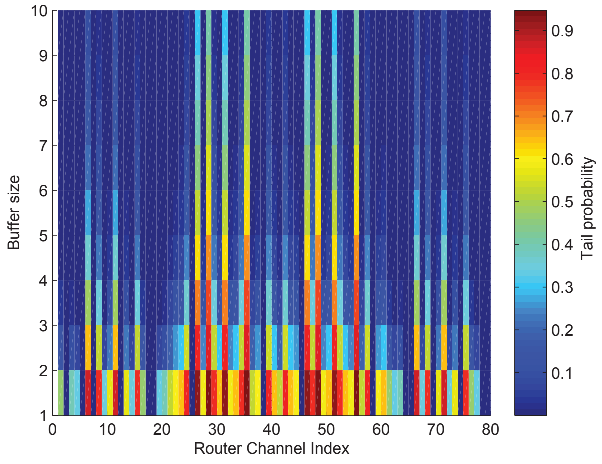


Figure 16. Buffer dimensioning analysis for 4x4 2D-mesh with uniform traffic at 0.31 packets/cycle.

uniform utilization of the buffers in the network. For this purpose, we repeat the buffer dimensioning analysis for the 4x4 2D-mesh applying uniform traffic. Again the injection rate was chosen close to the network saturation (0.31 packets/cycle at a basic flit service rate of $\mu = 0.5$). The results are depicted in Figure 16. We can see that the buffer utilization is now much more uniform than in the case of the application specific traffic. Nevertheless, we still find some bottlenecks in the region of buffer 26 to 35 and 46 to 55. Referring to Figure 12, we find that all these buffers are inputs of center routers. The center bottleneck is a commonly known characteristic of 2D-mesh topologies under uniform traffic.

The result confirms the validity and suitability of the proposed analytic model. It shows that we can already gain much insight into the system behavior in an early design stage with only a limited amount of information concerning system parameters.

D. DSE Scenario

Now, we have all necessary analytic tools at hand to clarify the question put at the end of the small DSE example in Section I. Therein, we proposed two alternative topologies (Figure 2(b) and Figure 2(c)) for a given application specific traffic scenario (Figure 2(a)). The first topology (solution a) employs two modules per router, while the second topology (solution b) spends a single router for every module. Finding a trade-off between latency, throughput and area consumption, we were not able to find a clear answer in Section I concerning which of the two alternatives is better suited. We investigated the small DSE example more closely performing a tail probability analysis using the proposed analytic model. Annotating the color-coded tail probabilities $P[x \geq 1]$ (according to Figure 12), we can visualize the performance bottlenecks for the two topologies, as depicted in Figure 17. It can be seen that the left router in solution a) is a serious bottleneck in the network. Indeed, the arriving traffic even exceeds the service capabilities of the router. Therefore, the average packet latency in the NoC tends to infinity, if we employ the infinite buffer queueing model. We can conclude that solution a) is not suited to fulfill the performance requirement of the given application scenario. Solution b) offers an increased bisection width. This results in a better spatial distribution of the traffic load in the network. Though the link between the routers at module 4 and 3 has still a high load, all routers are able to handle the incoming traffic. The average latency in the network is 8.4 cycles, according to the analytic results. Now, we are able to make a clear decision. Topology a) is not able to achieve the required throughput. Therefore, solution b) would be the better alternative in this case, even though the additional two routers cause increased chip area. The results of this small DSE demonstrate that a little change can sometimes make a big difference. Hence, it is worth to spend some effort to investigate the interconnection more closely.

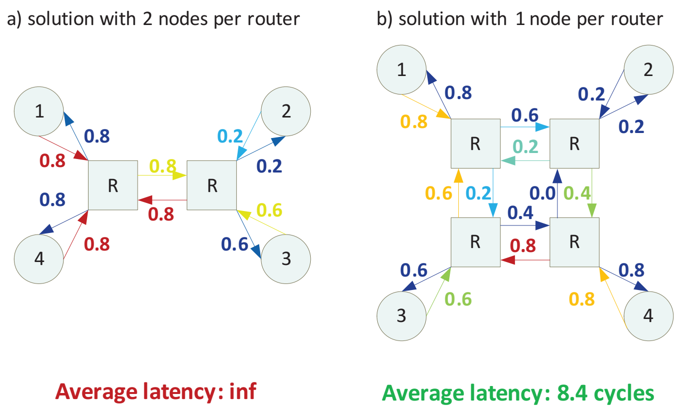

a) solution with 2 nodes per router
b) solution with 1 node per router
R
0.8
0.2
0.6
0.4
0.0
Average latency: inf
Average latency: 8.4 cycles

Figure 17. Results for small DSE example from Section I employing analytic model; comparison of solution with one (a) and two (b) modules per router. The color denotes the tail probabilities ($P[x \geq 1]$) at the corresponding input buffers (red=high, yellow=medium, blue=low).

VI. Extensions

The previous section demonstrated the feasibility and accuracy of the basic NoC model. However, several extensions are possible to further increase the accuracy and expand the application area of the proposed model.

A. Transient Behavior of NoC Router

The router and NoC models presented in Section IV describe the steady-state behavior of NoCs with adequate accuracy, but real systems do not necessarily work in steady-state. For instance, after starting a system, it needs time to converge to steady-state. Furthermore, application and communication patterns can change over time causing time-varying traffic rates in NoCs. An example of adaptive application and communication patterns can be found in [12], where a runtime adaptive NoC with dynamic routes and buffer allocations is proposed. This example illustrates that stationarity is not assured and it is of principal interest to consider transient characteristics as well. In the following, we outline model extensions that capture the transient behavior of a router.

In order to enable a transient analysis, we apply a generalization of the aggregation technique published in [31]. The technique in [31] is designed for continuous-time queueing systems, but we adapt it to discrete-time processes. Most variables used so far become time-dependent, e.g., arrival rates are described by $\lambda(t)$ instead of λ.

1) Transient behavior of uncoupled queues: The first step of the modified aggregation approach is to determine the transient behavior of uncoupled, independent queues described by the state probabilities $\pi_k(\boldsymbol{y}, t)$, which has to be done for all macro states $\boldsymbol{y}$ separately. We utilize standard tools for discrete-time markov chains for constructing a transition matrix Q [11]. The transient behavior of the state probabilities is computed iteratively by multiplying the state probabilities of the previous time step by the transition matrix, i.e., $\pi(\boldsymbol{y}, t+1) = \pi(\boldsymbol{y}, t)Q$. In order to enable this computation, we restrict the state space of a single queue to be finite.

2) Transient behavior of coupled queues: In the second part of the aggregation approach, the state probabilities of independent queues are utilized for approximating the transient behavior of coupled queues. Referring to [31], the transient rates among macro states (see (12)) can be generalized to

$$p(\boldsymbol{y}, \boldsymbol{y}', t) = \begin{cases} \lambda_i(t) & \text{for } \boldsymbol{y}' = \boldsymbol{y} + e_i, \\ (1-\lambda_i(t))\mu_i(\boldsymbol{y}, t)\frac{\pi_1(\boldsymbol{y},t)}{(1-\pi_0(\boldsymbol{y},t))} & \text{for } \boldsymbol{y}' = \boldsymbol{y} - e_i, \\ 0 & \text{else.} \end{cases}$$

Next, we summarize the transition rates to a transition matrix $\tilde{Q}(t)$. By iterative multiplication of the previous system state by the transition matrix, we derive the transient behavior of the coupled system, i.e., $\sigma(\boldsymbol{y}, t+1) = \sigma(\boldsymbol{y}, t)\tilde{Q}(t)$. Finally, in analogy to Section IV-C, the resulting macro state probabilities $\sigma(\boldsymbol{y}, t)$ can be used to determine various key performance indicators.

3) Numerical evaluation: For illustration, we consider a single router with three input/output pairs. Flits enter the router at one input and leave it at an output of a different input/output pair. Therefore, flits cannot be routed from the input to the output of the same input/output pair. We assume a maximum

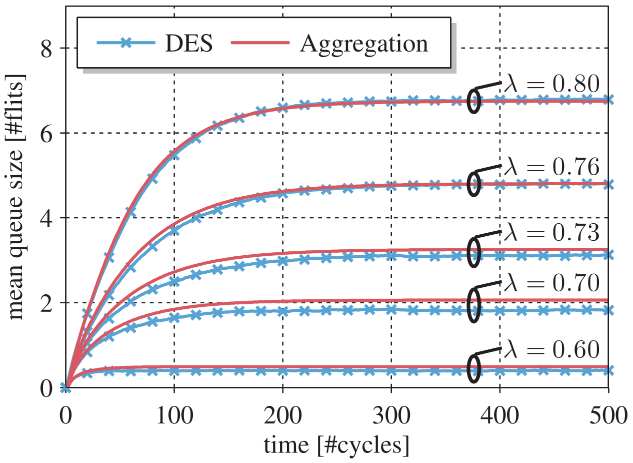


Figure 18. Mean queue length of one input of the router over time and after system startup. The proposed aggregation technique is compared to a discrete event simulation for different arrival rates λ.

finite buffer length of 10 flits per queue. At each input, an arrival process with a fixed mean intensity λ is assumed, and the traffic destinations are uniformly distributed. In different simulation runs, we use diverse mean intensities between 0.6 and 0.8 flits per cycle. If needed, time-varying traffic intensities can also be applied within one simulation run. We utilize the transient model and determine the average buffer load according to (13) over the first 500 cycles after system startup. The results in Figure 18 show that the time the system needs to converge to stationarity clearly depends on the arrival rate. For high traffic scenarios the system needs about 300 cycles, while the same system approaches stationarity for $\lambda = 0.6$ within 50 cycles.

Furthermore, we compare the aggregation technique to a cycle-accurate discrete event simulation (DES), where we simulate the system event by event. We perform a Monte Carlo simulation and average over 30000 realizations in order to obtain the average behavior of the system. The comparison of the aggregation technique to the DES reveals that the approximation works quite accurately, especially for low and high traffic scenarios the error is negligibly small. The largest error can be found for $\lambda = 0.7$, where the relative error of the mean queue size is 11%.

There are several applications for the transient model. It can, for instance, be used to predict dynamic behavior of NoC in algorithms that adapt application mappings or traffic patterns dynamically.

B. Blocking in NoC with finite buffers and generalized service

The model approach, proposed in Section IV-C, is based on an infinite buffer queueing model, which offers the advantage of low computational complexity. However, this assumption also entails some drawbacks. First, the spatial distribution of the traffic congestion in the network cannot be considered. Therefore, the prediction of performance measures becomes inaccurate, since every router is analyzed independently of the

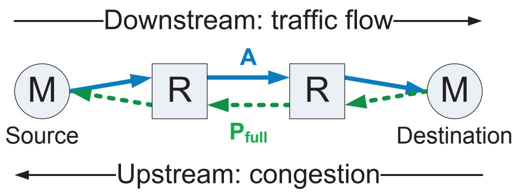


Figure 19. Mutual dependency of accepted traffic (A) and congestion feedback parameter P_{full} in finite buffer queueing networks.

subsequent (i.e., downstream) routers. This does not reflect the real on-chip situation accurately. Finally, buffer dimensioning based on tail probabilities, as demonstrated in Section V, is feasible but with limited accuracy. This is because blocking of injected traffic is not considered by the infinite buffer model.

Therefore, it is reasonable to consider an extension of the basic NoC model for finite buffer constraints. In the scope of the section, we sketch first ideas and challenges, which come with this extension. The integration and numerical evaluation is up for future work. Finite buffer models are well-known in queueing theory. Closed-form solutions are available for the most simple form, the M/M/1/K system [11]. However, first investigations show that the assumption of exponentially distributed service times limits the accuracy gain of the finite buffer model. Hence, it seems reasonable to consider a more general M/G/1/K system for this purpose. Unfortunately, there is no closed-form solution available for the distribution of the number of customers in an M/G/1 queueing system. A quite accurate and computationally efficient two-moment approximation is proposed in [32]. It can easily be extended to M/G/1/K systems following the approach of [33]. A good estimation of the blocking probability in an M/G/1/K system is also proposed by [34].

The biggest challenge in modeling finite buffer queueing networks is the spatial distribution of the traffic congestion. In case that an input buffer is filled, the service at the preceding (i.e., upstream) router stops for the corresponding output port. This means that the traffic congestion is propagated in upstream direction, the opposite direction of the data flow. It is necessary to derive an analytic expression for the probability that the buffer is filled, which is propagated as parameter in upstream direction to model traffic congestion. Note that this probability does not correspond to the blocking probability, since packets are never blocked (i.e., rejected) once they are injected in the NoC.

We realize that the accepted traffic (A) is propagated downstream while the congestion parameter P_{full} is propagated upstream, as depicted in Figure 19. Unfortunately, we find mutual dependencies between these two parameters in the network, if we consider general topologies. We propose an iterative fixed-point algorithm on network level to approximate the steady-state solution of a finite buffer queueing network.

We conclude that though this approach will become more computationally complex, it promises for an increased accuracy and enables analyzing network congestion and blocking of the injected traffic.

VII. Conclusion & Future Work

In this paper, we presented an analytic approach for modeling on-chip networks for many-core SoC based on queueing theory. In contrast to many existing models, the approach is very flexible in terms of supported topology, routing scheme and traffic pattern. The approach overcomes the limitations of the mean value analysis introduced in the existing work. Instead, it provides information about a steady-state distribution of the network routers. This allows to derive various key performance indicators, such as overflow probabilities or average queueing delays, which is very important information for dimensioning network resources, such as buffers, links, etc. We demonstrated the very high accuracy of the approach by comparison to a cycle-accurate simulation. The average estimation error for the mean latencies in a 4x4 2D-mesh is only 3%. The application of the proposed model was shown based on some simple design examples for buffer dimensioning, localizing bottlenecks, and benchmarking topologies. Extensiona of the basic model were proposed to consider finite buffers, dynamic traffic, and to generalize the service time assumptions made for the network routers. This further increases the accuracy of the basic analytic model and expands its application area.

The application and detailed evaluation of the suggested model extensions are left for future work. Furthermore, the consideration of link failures and error-prone routers employing a stochastic error model allows to investigate resiliency mechanisms for NoC. Finally, supporting multiple clock domains (i.e., globally asynchronous locally synchronous systems) and frequency scaling is another open topic in order to explore a many-core NoC more accurately.

Acknowledgment

This work was partly sponsored by the European Social Fund and the Free State of Saxony within the project *Secure Remote Execution* (SREX, nr 100111037).

References

[1] E. Fischer, A. Fehske, and G. Fettweis, “A Flexible Analytic Model for the Design Space Exploration of Many-Core Network-on-Chips Based on Queueing Theory,” in Proceedings of the Fourth International Conference on Advances in System Simulation, ser. SIMUL ’12, 2012, pp. 119–124.

[2] TU Dresden, “ESF Young Investigators Group; 3D Chip Stack Intraconnects - 3DCSI,” last visited on 12/12/2013. [Online]. Available: http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_elektrotechnik_und_informationstechnik/3dcsi

[3] J. Manferdelli, N. Govindaraju, and C. Crall, “Challenges and Opportunities in Many-Core Computing,” Proc. of IEEE, vol. 96, no. 5, May 2008, pp. 808 –815.

[4] S. Borkar, “Thousand Core Chips: A Technology Perspective,” in Proceedings of the 44th Annual Design Automation Conference, ser. DAC ’07, 2007, pp. 746–749.

[5] J. Nickolls and W. Dally, “The GPU Computing Era,” Micro, IEEE, vol. 30, no. 2, March-April 2010, pp. 56 –69.

[6] T. Limberg, M. Winter, M. Bimberg, R. Klemm, E. Matus, M. Tavares, G. Fettweis, H. Ahlendorf, and P. Robelly, “A fully programmable 40 GOPS SDR single chip baseband for LTE/WiMAX terminals,” in Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European, 2008, pp. 466–469.

[7] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in Design Automation Conference, 2001. Proceedings, 2001, pp. 684–689.

[8] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," Computer, vol. 35, no. 1, Jan 2002, pp. 70 –78.

[9] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, Jan. 2009, pp. 3 –21.

[10] J.-Y. Le Boudec and P. Thiran, Network calculus: a theory of deterministic queueing systems for the internet. Berlin, Heidelberg: Springer-Verlag, 2001.

[11] L. Kleinrock, Queueing systems - 1 : Theory. New York: Wiley, 1975.

[12] M. Al Faruque, T. Ebi, and J. Henkel, "AdNoC: Runtime Adaptive Network-on-Chip Architecture," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 2, 2012, pp. 257–269.

[13] P. Bogdan and R. Marculescu, "Non-Stationary Traffic Analysis and Its Implications on Multicore Platform Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, 2011, pp. 508–519.

[14] W. Dally, "Performance analysis of k-ary n-cube interconnection networks," IEEE Transactions on Computers, vol. 39, no. 6, Jun 1990, pp. 775 –785.

[15] A. E. Kiasari, D. Rahmati, H. Sarbazi-Azad, and S. Hessabi, "A Markovian Performance Model for Networks-on-Chip," in 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2008. PDP 2008., 2008, pp. 157–164.

[16] A. Kiasari, Z. Lu, and A. Jantsch, "An Analytical Latency Model for Networks-on-Chip," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, 2013, pp. 113–123.

[17] J. Hu, U. Ogras, and R. Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 12, 2006, pp. 2919–2933.

[18] N. Nikitin and J. Cortadella, "A Performance Analytical Model for Network-on-Chip with Constant Service Time Routers," in Proceedings of the 2009 International Conference on Computer-Aided Design, ser. ICCAD '09, 2009, pp. 571–578.

[19] U. Ogras, P. Bogdan, and R. Marculescu, "An Analytical Approach for Network-on-Chip Performance Analysis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 12, Dec. 2010, pp. 2001 –2013.

[20] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi, "Analytical modeling and evaluation of On-Chip Interconnects using Network Calculus," in 3rd ACM/IEEE International Symposium on Networks-on-Chip, ser. NoCS 2009, 2009, pp. 74–79.

[21] A. E. Kiasari, A. Jantsch, and Z. Lu, "Mathematical formalisms for performance evaluation of networks-on-chip," ACM Computing Surveys (CSUR), vol. 45, no. 3, 2013, p. 38.

[22] U. Ogras and R. Marculescu, "Target noc platform," in Modeling, Analysis and Optimization of Network-on-Chip Communication Architectures, ser. Lecture Notes in Electrical Engineering. Springer Netherlands, 2013, vol. 184, pp. 39–47.

[23] M. Bossert and M. Breitbach, Digitale Netze / Funktionsgruppen digitaler Netze und Systembeispiele. Stuttgart ; Leipzig: Teubner, 1999.

[24] E. Fischer and G. P. Fettweis, "An accurate and scalable analytic model for round-robin arbitration in network-on-chip," in Seventh IEEE/ACM International Symposium on Networks on Chip, ser. NoCS '13, 2013, pp. 1–8.

[25] Seber and A. F. George, A Matrix Handbook for Statisticians. John Wiley & Sons, Inc., 2008.

[26] R. Nelson, Probability, stochastic processes, and queueing theory / the mathematics of computer performance modeling. New York ; Heidelberg [u.a.]: Springer, 1995.

[27] A. Fehske and G. Fettweis, "Aggregation of variables in load models for interference-coupled cellular data networks," in IEEE International Conference on Communications (ICC), 2012, 2012, pp. 5102–5107.

[28] H. A. Simon and A. Ando, "Aggregation of Variables in Dynamic Systems," Econometrica, vol. 29, no. 2, Apr 1961, pp. 111–138.

[29] worm_sim, "Cycle-accurate noc simulator," last visited on 12/12/2013. [Online]. Available: https://www.ece.cmu.edu/~sld/software.html

[30] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 4, 2005, pp. 551–562.

[31] D. Öhmann, A. Fehske, and G. P. Fettweis, "Transient Flow Level Models for Interference-Coupled Cellular Networks," in 51st Annual Allerton Conference on Communication, Control, and Computing, Monticello, 2013.

[32] D. S. Myers and M. K. Vernon, "Estimating queue length distributions for queues with random arrivals," SIGMETRICS Perform. Eval. Rev., vol. 40, no. 3, Jan. 2012, pp. 77–79.

[33] J. Virtamo, "Queueing Theory; M/G/1-queue," 2013, lecture notes of Queuing theory, 38.3143, last visited on 12/12/2013. [Online]. Available: http://www.netlab.tkk.fi/opetus/s383143/kalvot/english.shtml

[34] J. M. Smith, "Properties and performance modelling of finite buffer M/G/1/K networks," Computers & Operations Research, vol. 38, no. 4, 2011, pp. 740 – 754.

Undecidable Case and Decidable Case of Joint Diagnosability in Distributed Discrete Event Systems

Lina Ye and Philippe Dague
INRIA, Grenoble-Rhône-Alpes, lina.ye@inria.fr
LRI, Univ. Paris-Sud, philippe.dague@lri.fr

Abstract—**Diagnosability is an important property that determines at design stage how accurate any diagnosis algorithm can be on a partially observable system. Most existing approaches assumed that each observable event in the system is globally observed. Considering the cases where there is no global information, one of our recent work proposed a new framework to check diagnosability in a system where each component can only observe its own observable events to keep the internal structure private in terms of observations. However, we assumed that the local paths in each component can be exhaustively enumerated, which is not suitable in a general case where there are embedded cycles. In this paper, we get some new results about diagnosability in such a system in a general case, i.e., what we call joint diagnosability in a self-observed distributed system. First, we prove the undecidability of joint diagnosability with unobservable communication events by reducing the Post's Correspondence Problem to joint diagnosability problem. We also propose an algorithm to check a sufficient but not necessary condition of joint diagnosability, which is then adapted when the assumption of all communication events being unobservable is relaxed, i.e., communication events could be either observable or unobservable. Then, we discuss about the decidable case where communication events are all observable and develop a new efficient algorithm to test it. Finally, we also provide an important property of joint diagnosability after analyzing its relationship with classical diagnosability.**

Keywords—joint diagnosability, self-observed systems, finite state machine, Post's Correspondence Problem, undecidability.

I. Introduction

OVER the latest decades, with the advancement of technologies, systems are becoming more and more complex since more performance requirements are imposed on them, which causes more errors that they are subject to. However, it is not realistic to manually detect faults for complex systems. Automated diagnosis mechanisms are therefore required to monitor large distributed applications such as transportation systems, communication networks, manufacturing systems, web services, spatial systems and power systems. Thus, it is crucial for a complex system to perceive that it is not operating correctly and then without human intervention, to detect and isolate original faults, which will be restored by repair plans to normality.

Generally speaking, diagnosis reasoning is to detect possible faults that can explain the observations. The possibility to achieve such a diagnosis reasoning depends on the diagnosability of the system. Diagnosability is an important property that determines at design stage how accurate any diagnosis algorithm can be on a partially observable system, which has significant economic impact on the improvement of performance and reliability of complex systems. The diagnosability analysis problem has received considerable attention in the literature, both in centralized and distributed ways.

In this paper, we study diagnosability problem for distributed systems where a component cannot observe the observable events in other components, which are called self-observed systems. We make several contributions by extending the conference article [1]. The first one is to extend diagnosability of globally observed systems to what we call joint diagnosability of self-observed systems before proving its undecidability in the case where communication events are unobservable. This is done by reducing the Post's Correspondence Problem to joint diagnosability problem, which is inspired from the undecidability result of joint observability [2]. The second one is to propose an algorithm for testing a sufficient condition of joint diagnosability with unobservable communication events. To do so, we first obtain pairs of local trajectories in the faulty component, such that for each pair, only one trajectory contains the fault but both trajectories have the same enough local observations. Their global consistency is then checked through two phases. We prove that it is a sufficient condition and point out why it is not necessary. Afterwards, we adapt this algorithm when the assumption of communication events being unobservable is relaxed, i.e., communication events could be observable or unobservable. The third one is to discuss about the decidable case where communication events are observable and to develop an efficient algorithm to test it. Finally, we provide an important property of joint diagnosability after analyzing its relationship with classical diagnosability.

This article extends our recent works [1] and [3] in the following aspects.

- We add detailed proofs for Theorem 1 and Lemma 2, which make our approach more convincing in terms of correctness.
- We define a new structure called local twin checker for a normal component to make the presentation more clear and the algorithm more efficient. The reason is that a local twin checker contains no fault information since it is constructed for a normal component without fault. Then we provide new complexity analysis for the proposed sufficient algorithm to test joint diagnosability when communication events are unobservable.
- We generalize the sufficient algorithm to test joint di-

agnosability when the assumption is relaxed such that communication events could be either observable or unobservable, which is presented in Section V-D.
- In Section VI, we provide more details about why joint diagnosability is decidable when communication events are all observable and then develop a new algorithm to test it.
- We provide a new important property of joint diagnosability after analyzing its relationship with classical diagnosability in Section VII.

The paper is organized as follows. In the next section, we talk about related works in the literature. In Section III, we model self-observed distributed systems and recall classical diagnosability and joint diagnosability. Section IV presents undecidability analysis for joint diagnosability when communication events are unobservable. An algorithm to test a sufficient condition is proposed in Section V with complexity analysis, which is then adapted when communication events could be either unobservable or observable. In Section VI, we discuss about decidable case for joint diagnosability, i.e., when communication events are observable, before giving an algorithm to test it. Finally, we compare joint diagnosability and classical diagnosability to find out their relationship in Section VII before giving the conclusion in Section VIII.

II. Related Work

In the literature, three types of systems are under investigation for diagnosis problem: continuous systems, discrete event systems and hybrid systems ([4], [5], [6], [7], [8], [9], [10], [11], [12]). In this paper, the dynamic systems studied are discrete event systems (abbreviated DES hereafter). Given a system, if its state space is naturally described by a discrete set and if state transitions only occur at discrete points in time, we associate these transitions with events and this system is called a DES. The reason why we choose DES for investigation is that most of the man-made systems are DES and that continuous systems can be abstracted to be DES. Nowadays, lots of works have been studied on control of DES, including diagnosis algorithm, diagnosability analysis, predictability analysis, etc. ([13], [14], [15], [16], [17], [18], [19]).

Some existing works analyzed diagnosability problem in a centralized way ([20], [21] and [22]), i.e., the knowledge of the monolithic model of a given system is hypothesized, which is the very powerful information for diagnosability analysis and leads to an unrealistic combinatorial explosion of the search space. This is why very recently distributed approaches for diagnosability began to be investigated ([23], [24] and [25]), relying on local objects. More precisely, in these distributed approaches, original diagnosability information can be obtained from the components where faults may occur and then the global decision is calculated by checking its global consistency. However, all these approaches assumed that each observable event in the system can be observed by all components, i.e., globally observed. However, in reality, there are some cases where it is not possible to obtain global information. For example, networked control systems are characterized by the fact that multiple distributed components possess their own

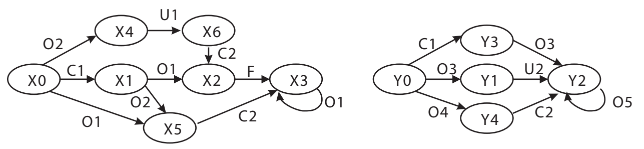


Fig. 1: A system with two components G_1 (left) and G_2 (right).

part of available information instead of a global knowledge. Then one of our recent work [26] proposed a new framework to check diagnosability in a system where each component can only observe its own observable events to keep the internal structure private in terms of observations. However, we assumed that the local paths in each component can be exhaustively enumerated, which is not suitable in a general case where there are embedded cycles. In this paper, we generalize this work to get some new results about the diagnosability of what we call self-observed distributed systems, i.e., systems where locally observable events can only be observed by their own component.

III. Preliminaries

In this section, we show how to model self-observed distributed DESs and then recall classical diagnosability and joint diagnosability.

A. System Model

We consider a self-observed distributed DES composed of a set of components $\{G_1, G_2, ..., G_n\}$ that communicate with each other by communication events. Each component can only observe its own observable events and thus keeps its internal structure private in terms of observations. Such a system is modeled by a set of finite state machines (FSM), each one representing the local model of one component.

Definition 1: (Local Model). The local model of a component G_i is a FSM, denoted by $G_i = (Q_i, \Sigma_i, \delta_i, q_i^0)$, where

- Q_i is the set of states;
- Σ_i is the set of events;
- $\delta_i \subseteq Q_i \times \Sigma_i \times Q_i$ is the set of transitions;
- q_i^0 is the initial state.

The set of events Σ_i is partitioned into four disjoint subsets: Σ_{i_o}, the set of locally observable events that can be observed only by their own component G_i; Σ_{i_u}, the set of unobservable normal events; Σ_{i_f}, the set of unobservable fault events; and Σ_{i_c}, the set of communication events shared by at least one other component, which are the only shared events between components. Figure 1 depicts a self-observed distributed system with two components: G_1 (left) and G_2 (right), where Oi denotes a locally observable event, F denotes an unobservable fault event, Ui denotes an unobservable normal event and Ci denotes a communication event.

We denote the synchronized FSM of components $G_1, ..., G_n$ by $\|(G_1, ..., G_n)$, where the synchronized events are the shared events between components and each one of them always occurs simultaneously in all components that define it. The state space of the synchronized FSM is the Cartesian product of the state spaces of components. The monolithic model (also called global model in the following) of the entire system is implicitly defined as the synchronized FSM of all components based on their shared events, i.e., communication events. However, the global model will not be calculated in this paper since in a self-observed distributed system, the global occurrence order of observable events is not accessible. In the following, we call the synchronization of a subset of components of G, i.e., $\|(G_{s_1}, ..., G_{s_m})$, as subsystem of G, where $\{s_1, ...s_m\} \subseteq \{1, ...n\}$. Note that one component or the entire system can also be considered as a subsystem.

Given a system model $G = (Q, \Sigma, \delta, q^0)$, the set of words produced by the FSM G is a prefix-closed language $L(G)$ that describes the normal and faulty behaviors of the system. Formally, $L(G) = \{s \in \Sigma^* | \exists q \in Q, (q^0, s, q) \in \delta\}$, where the transition δ has been extended from events to words. In the following, we call a word of $L(G)$ a **trajectory** in the system G and a sequence $q_0\sigma_0 q_1\sigma_1...$ a **path** in G, where $\sigma_0\sigma_1...$ is a trajectory and for all i, we have $(q_i, \sigma_i, q_{i+1}) \in \delta$. Given $s \in L(G)$, we denote the post-language of $L(G)$ after s by $L(G)/s$ and denote the projection of s to observable events of G (resp. G_i) by $P(s)$ (resp. $P_i(s)$). For example, if $s = O1.U2.O3^*$, then we have $P(s) = O1.O3^*$, where Oi denotes an observable event. These notations are also appropriate for local components. We adopt the assumption described in [23], i.e., the projection of the global language on each local model is observable live, in particular there is no unobservable cycle in any component. For the sake of simplicity, our approach is illustrated by dealing with only one fault, which can be extended to the case with multiple faults.

B. Diagnosability and Joint Diagnosability

We now recall classical diagnosability for centralized DESs. Informally speaking, the existence of two indistinguishable behaviors, i.e., holding the same enough observations with exactly one of them containing the given fault F, violates diagnosability property. The diagnosability approaches consist in checking the existence of such indistinguishable behaviors. In other words, a fault F is diagnosable in a system G iff its occurrence is determinable when enough events are observed from G after the occurrence of F, which is formally defined as follows [20], where s^F denotes a trajectory ending with F.

Definition 2: (Diagnosability). A fault F is diagnosable in a system G iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, |t| \geq k \Rightarrow$$
$$(\forall p \in L(G),\ P(p) = P(s^F.t) \Rightarrow F \in p).$$

The above definition states that if F is diagnosable, then for each trajectory s^F in G, for each t that is an extension of s^F with sufficient events, every trajectory p in G that is observation equivalent to $s^F.t$ should contain F. We call a pair of trajectories p and $p\prime$ satisfying the following conditions as a **critical pair**:

- p contains F and $p\prime$ does not;
- p has arbitrarily long events after the occurrence of F;
- $P(p) = P(p\prime)$.

It has been proved that the existence of critical pairs violates Definition 2 and thus witnesses non-diagnosability [21]. For distributed systems, the analysis of the above classical diagnosability requires global observations, which is not suitable for self-observed distributed systems where observable events can only be observed by their own component. Now we give the definition of joint diagnosability that only requires local observations without considering their global occurrence order [26].

Definition 3: (Joint diagnosability). A fault F is jointly diagnosable in a self-observed distributed system G composed of components $\{G_1, ...G_n\}$, iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, (\forall i \in$$
$$\{1, ..., n\}, |P_i(t)| \geq k) \Rightarrow (\forall p \in L(G)$$
$$(\forall i \in \{1, ..., n\}, P_i(p) = P_i(s^F.t)) \Rightarrow F \in p).$$

Joint diagnosability of a fault F means that for each trajectory s^F in G, for each t that is an extension of s^F with enough locally observable events in all components, every trajectory p in G that is equivalent to $s^F.t$ for local observations in each component should contain in it F. In a self-observed system, we call a pair of trajectories p and $p\prime$ satisfying the following conditions an (global) **indeterminate pair**:

- p contains F and $p\prime$ does not;
- p has arbitrarily long local observations in all components after the occurrence of F;
- $\forall i \in \{1, ..., n\}$, $P_i(p) = P_i(p\prime)$.

Here arbitrarily long local observations can be considered as infinite local observations. Now we have the following theorem [26].

Theorem 1: Given a self-observed distributed system G, a fault F is jointly diagnosable in G iff there is no (global) indeterminate pair in G.

Proof:

($\Rightarrow$) Suppose that F is jointly diagnosable in a system G and that there exists an indeterminate pair p and $p\prime$ with only p containing the fault F. Now let s^F denote the subpart of p that is ending with F and let t denote the rest part of p, i.e., $p = s^F.t$. Since p and $p\prime$ are an indeterminate pair, from its definition, we have that p has arbitrarily long local observations for each component G_i after the occurrence of F, and that for each component G_i, p and $p\prime$ have the same local observations, i.e., $P_i(p) = P_i(p\prime)$. However, $p\prime$ does not contain F. This contradicts the definition of joint diagnosability, where any trajectory with the same enough local observations in each component as $s^F.t$ should also contain F. So F is not jointly diagnosable in G, which contradicts the assumption.

($\Leftarrow$) Now suppose that there is no indeterminate pair in G and F is not jointly diagnosable in G. From the non joint diagnosability of F and Definition 3, we deduce that $\exists s^F \in L(G)$, such that there exists at least one extension t with enough local observations (represented by infinite paths with cycles) in all components, for which there must exist at least one other trajectory p containing the same local observations as $s^F.t$ for each component but without fault. Since the enough local observations of $s^F.t$ and p come from infinite paths with cycles, $s^F.t$ and p can be prolonged arbitrarily long. It follows that $s^F.t$ and p are an indeterminate pair since they satisfy three conditions of the definition of an indeterminate pair,

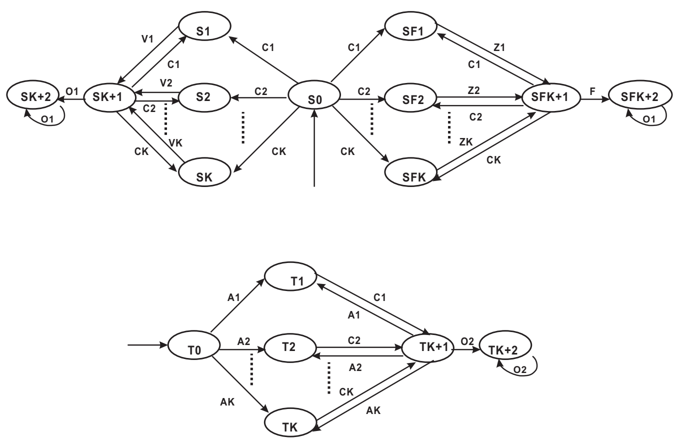


Fig. 2: A system with two components G_1 (top) and G_2 (bottom) for proving undecidability of joint diagnosability.

which contradicts the assumption that there is no indeterminate pair. ■

Another important notation that will be used in joint diagnosability analysis is reconstructibility [27], which is rephrased as follows.

Definition 4: (Reconstructibility). Given a system G that is composed of several subsystems, i.e., $G = \|(G_{s_1}, ..., G_{s_m})$, let ρ be a trajectory in G, the set of trajectories obtained by projecting ρ on these subsystems, i.e., denoted by $\rho_1 = P'_{G_{s_1}}(\rho), ..., \rho_m = P'_{G_{s_m}}(\rho)$, where $P'_{G_{s_i}}(\rho)$ is the projection of ρ on the subsystem G_{s_i}, is said to be reconstructible with respect to G.

Lemma 1: In a system G, given two subsystems G_S and G'_S, if $\Sigma_{S_c} \cap \Sigma_{S'_c} = \emptyset$, then $\forall(s, s\prime), s \in L(G_S), s\prime \in L(G'_S)$, s and $s\prime$ are reconstructible.

Lemma 1 means that if there is no common communication event between two subsystems, then any trajectory in one subsystem and any one in the other one are reconstructible.

IV. Undecidable Case

To discuss about joint diagnosability, we consider two cases: communication events being unobservable and observable. In this section, we first consider the case where all communication events are unobservable.

Theorem 1 implies that checking joint diagnosability boils down to check the existence of indeterminate pairs that witnesses non joint diagnosability. Before discussing about how to test this, it is very important to check whether the problem is decidable or not. Inspired from [2], where undecidability of joint observability is proved by reducing the Post's Correspondence Problem (PCP) to an observation problem, we now prove that joint diagnosability is not decidable with communication events being unobservable.

Theorem 2: Given a self-observed distributed system where communication events are unobservable, checking joint diagnosability of a given fault is undecidable.

Proof:

1) PCP: given a finite alphabet Σ, two sets of words $v_1, v_2, ..., v_k$ and $z_1, z_2, ..., z_k$ over Σ, then a solution to PCP is a sequence of indices $(i_m)_{1\leq m\leq n}$ with $n \geq 1$ and $1 \leq i_m \leq k$ for all m such that $v_{i_1}v_{i_2}...v_{i_n} = z_{i_1}z_{i_2}...z_{i_n}$.

2) Now consider the example depicted in Figure 2, where the system is composed of two components G_1 and G_2. In G_1, each one of $Vi, i \in \{1, ..., k\}$, or each one of $Zi, i \in \{1, ..., k\}$, denotes a sequence of observable events all different from the observable event $O1$, and $C1, ..., Ck$ are unobservable communication events, then F denotes a fault event. In G_2, each one of $Ai, i \in \{1, ..., k\}$, denotes an observable event different from the observable event $O2$, and $C1, ..., Ck$ are unobservable communication events. Then the observations in G_1 can be described as $Vi_1Vi_2...Vi_nO1^*$ without fault or $Zi_1Zi_2...Zi_nO1^*$ with fault, where $\forall i_j, j \in \{1, ..., n\}, i_j \in \{1, ..., k\}$. In G_2, the observations are $Ai_1Ai_2...Ai_nO2^*$.

3) With the observation of $O1$, the local observations are $wO1^+$ for G_1 and $Ai_1Ai_2...Ai_nO2^*$ for G_2, where $w = Vi_1Vi_2...Vi_n$ when there is no fault or $w = Zi_1Zi_2...Zi_n$ when there is a fault. Clearly, if PCP has a solution, i.e., $\exists(i_m)_{1\leq m\leq n}$ such that $Vi_1Vi_2...Vi_n = Zi_1Zi_2...Zi_n$, we have two trajectories p and $p\prime$ such that the observations of p in G_1 are $Vi_1Vi_2...Vi_nO1^+$, which is a trajectory without fault, while the observations of $p\prime$ in G_1 are $Zi_1Zi_2...Zi_nO1^+$, which is a trajectory with a fault. And both p and $p\prime$ have the same observations for G_2, i.e., $Ai_1Ai_2...Ai_nO2^*$. Thus we get that p and $p\prime$ have the same observations for both G_1 and G_2, i.e., $Vi_1Vi_2...Vi_nO1^+$=$Zi_1Zi_2...Zi_nO1^+$ for G_1 and $Ai_1Ai_2...Ai_nO2^*$ for G_2, then the fault is not jointly diagnosable.

4) On the other hand, if the fault is not jointly diagnosable, then we obtain at least one indeterminate pair, denoted by p and $p\prime$ such that the projection of p on G_1 is $Ci_1Vi_1Ci_2Vi_2...Ci_nVi_nO1^*$, on G_2 is $Ai_1Ci_1Ai_2Ci_2...Ai_nCi_nO2^*$ and that of $p\prime$ on G_1 is $Cj_1Zj_1Cj_2Zj_2...Cj_mZj_mFO1^*$ and on G_2 is $Aj_1Cj_1Aj_2Cj_2...Aj_mCj_mO2^*$. From the fact that p and $p\prime$ have the same observations for G_2, we get $Ai_1Ai_2...Ai_nO2^* = Aj_1Aj_2...Aj_mO2^*$ and thus we have $m = n$ and $i_1 = j_1, ..., i_n = j_n$. And then from the same observations of p and $p\prime$ on G_1, we get $Vi_1Vi_2...Vi_nO1^* = Zi_1Zi_2...Zi_nO1^*$, i.e., $Vi_1Vi_2...Vi_n = Zi_1Zi_2...Zi_n$, which means that there is a solution for PCP.

5) From above, we get that checking the existence of a solution for PCP is equivalent to checking the existence of an indeterminate pair. Since PCP is an undecidable problem, checking joint diagnosability is also undecidable. ■

There are two major differences between joint diagnosability in our framework and joint observability in [2]. One is that the former assumes that local observers are attached to local components that are synchronized by common communication events to get a global model while the latter separates arbitrarily the observable events in the global model into several sets. The other one is that joint diagnosability consists in separating infinite trajectories while joint observability consists in separating finite ones. Thus, if any communication event is assumed to be unobservable, joint diagnosability checking boils down to infinite PCP. But this one has been proved to be undecidable [28].

V. Algorithm to Test a Sufficient Condition

We have proved that joint diagnosability in self-observed distributed systems with unobservable communication events is undecidable. We can nevertheless propose an algorithm to test a sufficient condition, which is still quite useful in some circumstances. The first step is to construct the local diagnoser for the faulty component, which allows one to get fault information after any local trajectory. Then we show how to build the corresponding local twin plant based on the local diagnoser to obtain the original information about indeterminate pairs (also called local indeterminate pairs in the following). The next step is to check the global consistency, i.e., whether the local indeterminate pairs can be extended into (global) indeterminate pairs, whose existence verifies non joint diagnosability. We give an algorithm to test a sufficient but not necessary condition for global consistency.

A. Original Diagnosability Information

From Theorem 1, we know that joint diagnosability verification consists in checking the existence of indeterminate pairs in the system. In the distributed framework, we use the structure called local twin plant, initially defined in [21], to analyze joint diagnosability. In particular, the considered fault is assumed to only occur in one component, denoted by G_F. Then the local twin plant for G_F contains original information for indeterminate pairs: this twin plant is a FSM that compares every pair of local trajectories to search for the pairs with the same arbitrarily long local observations, but exactly one of the two containing a fault, which are called local indeterminate pairs. First, we define an operation called delay closure with respect to a subset Σ_d of Σ to preserve all information about the events in Σ_d by abstracting away other parts.

Definition 5: (Delay Closure). Given a FSM G = (Q, Σ, δ, q^0), its delay closure with respect to $\Sigma_d \subseteq \Sigma$ is $\complement_{\Sigma_d}(G) = (Q, \Sigma_d, \delta_d, q^0)$ where $(q, \sigma, q') \in \delta_d, \sigma \in \Sigma_d$ iff $\exists s \in (\Sigma \backslash \Sigma_d)^*, (q, s\sigma, q') \in \delta$.

We now describe how to construct the local diagnoser for a given component, based on which we build the local twin plant.

Definition 6: (Local Diagnoser). Given a component model G_i, its local diagnoser is the nondeterministic FSM $D_i = (Q_{D_i}, \Sigma_{D_i}, \delta_{D_i}, q^0_{D_i})$, where $Q_{D_i} \subseteq Q_i \times 2^{\Sigma_{i_f}}$ is the set of states, $\Sigma_{D_i} = \Sigma_{i_c} \cup \Sigma_{i_o}$ is the set of events, $\delta_{D_i} \subseteq Q_{D_i} \times \Sigma_{D_i} \times Q_{D_i}$ is the set of transitions, and $q^0_{D_i} = (q^0_i, N)$ is the initial state of the diagnoser. The transitions of δ_{D_i} are those $((q, l), e, (q\prime, l\prime))$ with (q, l) reachable from the initial state $q^0_{D_i}$, and satisfying the following condition: there is a transition path $p = (q \xrightarrow{u_1} q_1 ... \xrightarrow{u_m} q_m \xrightarrow{e} q\prime)$ in G, with $u_k \in \Sigma_{i_u} \cup \Sigma_{i_f}, \forall k \in \{1, ..., m\}$, $e \in \Sigma_{i_o} \cup \Sigma_{i_c}$ and $\ell\prime = \ell \cup (\{u_1, ... u_m\} \cap \Sigma_{i_f})$.

Without loss of generality, we give the definition of local diagnoser for the set of faults in the component G_i, which is perfectly suitable to deal with single fault when we run our algorithm each time for one fault since twin plant method has exponential complexity with the number of faults. A diagnoser constructed as above shows fault information after any sequence of communication events and observable events in the faulty component. Figure 3 shows the local diagnoser

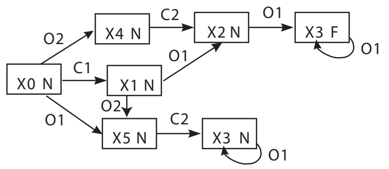


Fig. 3: Local diagnoser for G_1 of the system in Figure 1.

for the component G_1 of the system depicted in Figure 1. Given a diagnoser, its corresponding twin plant is obtained by synchronizing this diagnoser with itself, called left and right instances (denoted by D^l_i and D^r_i, respectively), based on the set of observable events. From Definition 6, we know that a diagnoser keeps observable events as well as communication events. However, since the local twin plant is to obtain all pairs with the same observations to search for local indeterminate pairs, only observable events should be synchronized. Since shared events are the only synchronized events, the non-synchronized events are distinguished between the two instances by the prefix L (R): in D^l_i (D^r_i), each communication event $c \in \Sigma_{i_c}$ from D_i is renamed by $L : c$ $(R : c)$. The names of all locally observable events remain the same. The definition of local twin plant is shown as follows.

Definition 7: (Local Twin Plant). Given a diagnoser D_i, the corresponding local twin plant is obtained by synchronizing its left instance with its right instance based on the set of observable events, denoted by $T_i = (Q_{T_i}, \Sigma_{T_i}, \delta_{T_i}, q^0_{T_i}) = D^l_i \| D^r_i$. Each state of a local twin plant is a pair of local diagnoser states providing two possible diagnoses with the same local observations. Given a local twin plant state $((q^l, l^l)(q^r, l^r))$, where (q^l, l^l) and (q^r, l^r) are two diagnoser states and each one contains both a system state and a fault label. If the considered fault $F \in l^l \cup l^r$ but $F \notin l^l \cap l^r$, which means that the occurrence of F is not certain up to this state, then this state is called an ambiguous state with respect to the fault F. An ambiguous state cycle is a cycle containing only ambiguous states. In a local twin plant, if a path contains an ambiguous state cycle with at least one locally observable event, then it is called a **local indeterminate path**, which corresponds to a local indeterminate pair. Note that local indeterminate paths contain original diagnosability information and can be obtained only in the local twin plant of the component G_F. If a local indeterminate pair can be extended into a global indeterminate pair, then we say that its corresponding local indeterminate path is globally consistent. Figure 4 presents the left and right instances of the local diagnoser for the component G_1 (top left and top right) as well as part of the corresponding local twin plant (bottom). The gray node represents an ambiguous state since the fault label is only contained in one of two diagnoser states. Thus, we can see that the corresponding path in this local twin plant is a local indeterminate path since it contains an ambiguous state cycle with a locally observable event.

We know that the function of the local twin plant is to obtain the original diagnosability information, i.e., all local indeterminate paths. From Figure 4, we can see that a local twin plant constructed as described above normally has a large

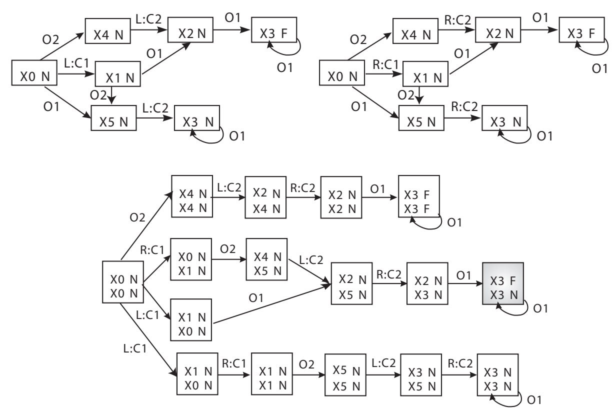


Fig. 4: Two instances of the diagnoser for G_1 (top) and part of the corresponding local twin plant (bottom).

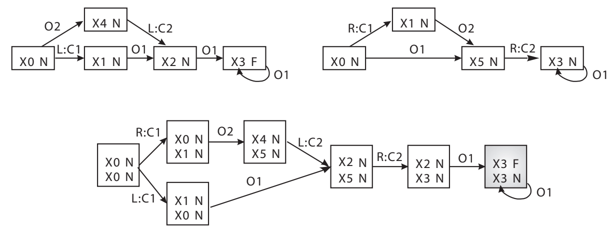


Fig. 5: Two reduced instances of the diagnoser for G_1 (top) and part of the corresponding reduced local twin plant (bottom).

redundant part, which is useless for a global decision, e.g., all paths without ambiguous state cycles. Furthermore, there could be several local indeterminate paths that correspond to the same pair of local trajectories in the local diagnoser. To simplify the local twin plant, we propose one way to reduce the two instances of the local diagnoser: D_i^l is obtained by retaining only paths with at least one fault cycle and D_i^r is reduced by retaining only paths with at least one cycle without fault state. This reduction keeps all necessary original diagnosability information since what we are interested in here are only local indeterminate pairs. Figure 5 shows the left and right reduced instances of the local diagnoser for the faulty component G_1 of Figure 1 (top) as well as part of the reduced local twin plant (bottom) that corresponds to the part of non-reduced local twin plant shown in the bottom part of Figure 4. In this way, the state space of the local twin plant constructed from reduced instances can be considerably reduced, which is called reduced local twin plant.

Lemma 2: The local twin plant corresponds to the same set of local indeterminate pairs as the reduced local twin plant.

Proof:
As described before, the local twin plant is constructed by synchronizing the non-reduced left and non-reduced right instances of the local diagnoser while the reduced one is constructed by synchronizing the reduced left and reduced right instances. For the sake of clarity, in the non-reduced left instance, we denote the set of paths with only fault state cycles by Λ^l_f, the set of paths without fault state cycle by $\Lambda^l_{\neg f}$, and then the set of paths with both fault state cycles and cycles without fault state by Λ^l_{both}. Similarly, in the non-reduced right instance, we denote the set of paths with only fault state cycles by Λ^r_f, the set of paths without fault state cycle by $\Lambda^r_{\neg f}$, and the set of paths with both fault state cycles and cycles without fault state by Λ^r_{both}. Then the local twin plant is constructed by $(\Lambda^l_{\neg f}\cup\Lambda^l_f\cup\Lambda^l_{both}) \parallel (\Lambda^r_{\neg f}\cup\Lambda^r_f\cup\Lambda^r_{both})$. On the other hand, the reduced left instance retains the paths with at least one fault state cycle and the reduced right instance retains the paths with at least one cycle without fault state. Then the reduced local twin plant is constructed by $(\Lambda^l_f\cup\Lambda^l_{both}) \parallel (\Lambda^r_{\neg f}\cup\Lambda^r_{both})$. The local twin plant construction can be expressed by the addition of the synchronized results of nine cases: 1) $(\Lambda^l_{\neg f}) \parallel_{\Sigma_{i_o}} (\Lambda^r_{\neg f})$; 2)$(\Lambda^l_{\neg f}) \parallel_{\Sigma_{i_o}} (\Lambda^r_f)$; 3)$(\Lambda^l_{\neg f}) \parallel_{\Sigma_{i_o}} (\Lambda^r_{both})$ 4)$(\Lambda^l_f) \parallel_{\Sigma_{i_o}} (\Lambda^r_{\neg f})$; 5) $(\Lambda^l_f) \parallel_{\Sigma_{i_o}} (\Lambda^r_f)$; 6) $(\Lambda^l_f) \parallel_{\Sigma_{i_o}} (\Lambda^r_{both})$; 7)$(\Lambda^l_{both}) \parallel_{\Sigma_{i_o}} (\Lambda^r_{\neg f})$; 8) $(\Lambda^l_{both}) \parallel_{\Sigma_{i_o}} (\Lambda^r_f)$ and 9) $(\Lambda^l_{both}) \parallel_{\Sigma_{i_o}} (\Lambda^r_{both})$. In the same way, the reduced local twin plant construction can be expressed by the addition of the synchronized results of four cases in the above, which are (case 4 + case 6 + case 7+ case 9). So compared to the reduced local twin plant, the local twin plant has five more synchronized results (case 1 + case 2 + case 3 + case 5 + case 8). Now consider case 4 and case 2, which are actually symmetrical. We can see that the part in left instance of case 2 is the same as the part in right instance of case 4 and the part in right instance of case 2 is the same as the part in left instance of case 4. It is easy to prove that the synchronized result of case 2 corresponds to the same set of local trajectory pairs in the local diagnoser as the synchronized result of case 4 does. In the same way, case 6 has the same result as case 8 and case 7 has the same result as case 3. Now the local twin plant has two more synchronized results (case 1 + case 5) than the reduced local twin plant. However, case 1 and case 5 can never get any local indeterminate pair. The reason is that in case 1, any path in $\Lambda^l_{\neg f}$ and in $\Lambda^r_{\neg f}$ has no fault state cycle, then the synchronized result has no ambiguous state cycle, which means that there is no corresponding local indeterminate pair. And in case 5, any path in Λ^l_f and in Λ^r_f has only fault state cycles. So their synchronization cannot obtain local indeterminate pair. Now we can say that the local twin plant corresponds to the same set of local indeterminate pairs as the reduced local twin plant does, which proves Lemma 2. ■

In our approach, we only construct the reduced local twin plant in the following, which is, from now on, called local twin plant for the sake of simplicity.

B. Global Consistency Checking

We obtain the set of local indeterminate paths in the local twin plant for the faulty component. However, up to now, its communication with the neighborhood in the whole system is not yet taken into account. Recall that if a local indeterminate pair can be extended into a global indeterminate pair, its corresponding local indeterminate path is said to be globally

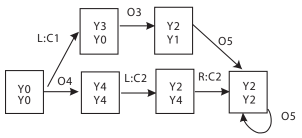


Fig. 6: Part of local twin checker for G_2 of the system in Figure 1.

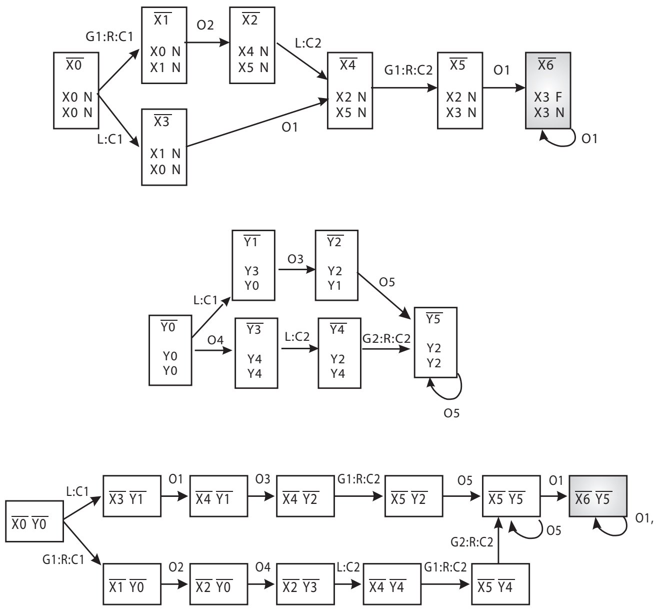


Fig. 7: Part of the renamed local twin plant for G_1 (top), part of renamed local twin checker for G_2 (middle) and part of the left consistent plant T_f^l (bottom).

consistent. In other words, joint diagnosability verification consists in checking the existence of globally consistent local indeterminate paths, whose existence proves non joint diagnosability. To check the global consistency, we consider now two important issues:

- the global consistency of the corresponding left trajectories of the local indeterminate paths in the local twin plant, shortly called left consistency checking in the following;
- the global consistency of the corresponding right trajectories of the local indeterminate paths in the local twin plant, shortly called right consistency checking.

Now we define another structure called local twin checker for a normal component, which is used to be synchronized with local twin plant for checking the global consistency. With similar idea as local twin plant, local twin checker is to obtain all pairs of local trajectories with the same observations, which has no fault information since it is defined for a normal component. Given a normal component, we first perform delay closure to keep only observable events and communication events. Then the left instance of the component is obtained by adding the prefix of L to non-synchronized events, i.e., communication events, and the right instance is acquired by adding the prefix of R to communication events, denoted by G_i^l and G_i^r, respectively.

Definition 8: (Local twin checker). Given a normal component G_i, its local twin checker is the FSM $C_i = G_i^l \| G_i^r$.
Figure 6 shows a part of the local twin checker for the component G_2 of the system depicted in Figure 1, where there is no fault information. To check left consistency and right consistency of local indeterminate paths, we define two consistent structures as follows.

Definition 9: (Left (Right) consistent plant). Given a subsystem G_S composed of $G_{i_1}, ..., G_{i_m}$, including the faulty component, and their corresponding local twin plant and local twin checkers, to obtain a left (right) consistent plant with respect to the subsystem G_S, denoted by T_f^l (T_f^r), we perform the following two steps:

- Distinguish right (left) communication events between local twin plant and local twin checkers by renaming them with the prefix of component ID. For example, $R{:}C2$ $(L{:}C2)$ in the local twin checker of G_2 is renamed as $G_2{:}R{:}C2$ $(G_2{:}L{:}C2)$.
- Note that observable events do not intersect between components and non-synchronized right (left) communication events are distinguished by the prefix of component ID. Now the renamed local twin plant and local twin checkers are synchronized, where the synchronized events are the common left (right) communication events.

In the left (right) consistent plant with respect to a subsystem G_S, each path p corresponds to a set of paths $p_{i_1}, ..., p_{i_m}$ in the local twin plant and local twin checkers of all components in G_S such that the set of left (right) trajectories of $p_{i_1}, ..., p_{i_m}$ are reconstructible with respect to G_S. Given a state in a left or right consistent plant, if it contains an ambiguous state in the local twin plant for the faulty component, then this plant state is also called an ambiguous state. For our example, the bottom part of Figure 7 shows a part of the left consistent plant T_f^l, which is obtained by synchronizing the renamed local twin plant of G_1 and the renamed local twin checker of G_2 (top and middle parts of Figure 7) based on the common left communication events. In the bottom part of this figure, the gray node $(\overline{X6Y5})$ is an ambiguous state of the plant since it contains the local twin plant state $\overline{X6}$ that is an ambiguous one.

C. Algorithm

Algorithm 1 presents the procedure to verify a sufficient condition of joint diagnosability. As shown in the pseudo-code, algorithm 1 performs as follows. Given the set of component models as well as the fault F that may occur in the component G_F as input, we initialize the parameters as empty, i.e., G_S^l, the subsystem for the left consistency checking and G_S^r, the subsystem for the right consistency checking. The procedure of the algorithm can be separated into two parts: left consistency checking and right consistency checking.

Algorithm 1 Algorithm to check a sufficient condition of joint diagnosability

```
1: INPUT: the system model $G = (G_1, ..., G_n)$; the fault $F$ and the faulty component $G_F$
2: Initializations: $G^l_S \leftarrow \emptyset$ (subsystem for left consistency checking); $G^r_S \leftarrow \emptyset$ (subsystem for right consistency checking)
3: $T^l_f \leftarrow ConstructLTP(G_F)$
4: $G^l_S \leftarrow G_F$
5: while $T^l_f \neq \emptyset$ and $DirectCC(G, G^l_S) \neq \emptyset$ do
6:     $G_i \leftarrow SelectDirectCC(G, G^l_S)$
7:     $C_i \leftarrow ConstructLTC(G_i)$
8:     $T^l_f \leftarrow T^l_f \| C_i$
9:     $G^l_S \leftarrow Add(G^l_S, G_i)$
10:    $T^l_f \leftarrow RetainConsisPaths(T^l_f)$
11: end while
12: if $T^l_f = \emptyset$ then
13:    return "$F$ is jointly diagnosable in $G$"
14: else
15:    $T^r_f \leftarrow AbstractRight(G_F, T^l_f)$
16:    $G^r_S \leftarrow G_F$
17:    while $T^r_f \neq \emptyset$ and $G^l_S \neq G^r_S$ do
18:        $G_i \leftarrow SelectDirectCC(G^l_S, G^r_S)$
19:        $T^r_f \leftarrow T^r_f \| AbstractRight(G_i, T^l_f)$
20:        $G^r_S \leftarrow Add(G^r_S, G_i)$
21:        $T^r_f \leftarrow RetainConsisPaths(T^r_f)$
22:    end while
23:    if $T^r_f = \emptyset$ then
24:        return "$F$ is jointly diagnosable in $G$"
25:    else
26:        return "Joint diagnosability cannot be determined"
27:    end if
28: end if
```

- Left consistency checking begins with the local twin plant construction of G_F, the subsystem G^l_S being now G_F (line 3-4). As long as both the left consistent plant T^l_f with respect to the current left subsystem G^l_S and $DirectCC(G,\ G^l_S)$ are not empty (line 5), where $DirectCC(G, G^l_S)$ is the set of directly connected components to the subsystem G^l_S (a directly connected component being one sharing at least one common communication event with the subsystem) the algorithm repeatedly performs the following steps to further check left consistency in an extended subsystem:
    1) Select one directly connected component G_i to the subsystem G^l_S and construct its local twin checker C_i (line 6-7).
    2) Synchronize T^l_f with C_i to obtain left consistent plant for this extended subsystem based on the set of common left communication events (line 8). To do so, the set of non-synchronized right communication events are distinguished by the prefix of component ID.
    3) Update the subsystem G^l_S by adding G_i and reduce the newly obtained T^l_f by retaining only paths with ambiguous state cycles containing observable events for all components in G^l_S, which are called consistent paths with respect to G^l_S (line 9-10).

If the left consistent plant T^l_f is empty, then there is no local indeterminate path that corresponds to a set of paths in the local twin plant and local twin checkers of all components in the subsystem such that their corresponding left trajectories are reconstructible. This follows that there is no globally consistent local indeterminate path. In this case, joint diagnosability information is returned (line 12-13). Otherwise, if T^l_f is not empty (line 14), then we proceed to check right consistency of the corresponding paths in T^l_f that have already been verified to be left consistent in the whole system.

- Right consistency checking begins with the function $AbstractRight(G_F,\ T^l_f)$ (line 15), which performs delay closure with respect to the set of right communication events and observable events of G_F. In this way, what we obtain does not contain left communication events, which are useless for right consistency checking. Then the subsystem G^r_S is assigned as G_F (line 16). As long as the right consistent plant T^r_f for the current right subsystem G^r_S is not empty and $G^l_S \neq G^r_S$ (line 17), we repeatedly perform the following steps to check right consistency in an extended subsystem (note that since left consistency checking does explore all connected components, during right consistency checking, we only need to consider the subsystem G^l_S instead of the whole system):
    1) Select a directly connected component G_i to G^r_S from G^l_S (line 18).
    2) Perform the function $AbstractRight(G_i,\ T^l_f)$, which has already been described above, and then synchronize with T^r_f based on the set of common right communication events (line 19). To do this, we rename the right communication events by removing the prefix of component ID, e.g., G_i:R:$C2$ renamed as R:$C2$.
    3) Update the subsystem G^r_S by adding G_i and reduce the newly obtained T^r_f by retaining only paths with ambiguous state cycles containing observable events for all components in G^r_S, i.e., consistent paths (line 20-21).

If the right consistent plant T^r_f is empty, then there is no local indeterminate path that corresponds to a set of paths in the local twin plant and local twin checkers such that their left trajectories and right trajectories are reconstructible respectively, i.e., there is no globally consistent local indeterminate path. In this case, the algorithm returns joint diagnosability information (line 23-24). Otherwise, if T^r_f is not empty, we cannot determine whether the fault is jointly diagnosable or not. Then the algorithm returns the information about

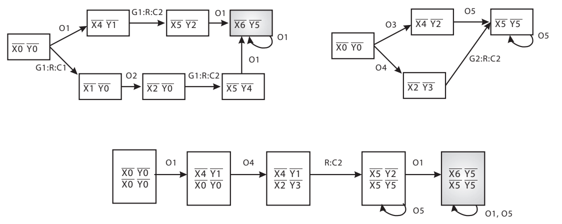


Fig. 8: FSM after delay closure on the left consistent plant (Figure 7) for G_1 (top left) and for G_2 (top right) and part of the right consistent plant (bottom).

the indetermination of joint diagnosability (line 25-26). In other words, empty left consistent plant T_f^l or empty right consistent plant T_f^r is a sufficient but not a necessary condition of joint diagnosability.

Theorem 3: In algorithm 1, if the left consistent plant T_f^l or the right consistent plant T_f^r is empty, then the fault is jointly diagnosable, but the reverse is not true.

Proof:
($\Rightarrow$) Suppose that T_f^l or T_f^r is empty and that the fault is not jointly diagnosable. From non joint diagnosability, it follows that there exists at least one globally consistent local indeterminate path. Since global consistency of a local indeterminate path implies both left consistency and right consistency, from algorithm 1, we know that, after left and right consistency checking, this local indeterminate path must correspond to a path both in T_f^l and in T_f^r. Thus, neither T_f^l nor T_f^r is empty, which contradicts the assumption.
($\not\Leftarrow$) Now we explain why if both T_f^l and T_f^r are not empty, it does not necessarily imply that the fault is not jointly diagnosable. Suppose that the left consistent plant T_f^l is not empty and that it contains two paths, denoted by $\rho 1$ and $\rho 2$, corresponding to two local indeterminate paths. $\rho 1$ corresponds to a set of paths $\rho_i^1, 1 \leq i \leq n$ in the local twin plant and local twin checkers of all components and $\rho 2$ corresponds to a set of paths $\rho_i^2, 1 \leq i \leq n$. Note that since the two sets of paths come from left consistent plant, then it is only guaranteed that the corresponding left trajectories of each set are reconstructible. Now suppose that the right trajectories of the set of paths $\rho_i^1, 1 \leq i \leq n$ are not reconstructible and the same for that of the set of paths $\rho_i^2, 1 \leq i \leq n$. It follows that the two local indeterminate paths cannot be extended into global indeterminate pairs and thus are not globally consistent. Then we further suppose that the right trajectories of the set of paths $\rho_1^1, ..., \rho_{n-1}^1, \rho_n^2$ are reconstructible and the same for the set of paths $\rho_1^2, ..., \rho_{n-1}^2, \rho_n^1$. From algorithm 1, it follows that finally the right consistent plant T_f^r is not empty. In this case, both T_f^l and T_f^r are not empty but there is no globally consistent local indeterminate paths, i.e., the fault is jointly diagnosable. ■

Now illustrate on our example that this condition is not necessary. The top parts of Figure 8 show the results of performing delay closure with respect to the set of right communication events and locally observable events both for G_1 and G_2 on the left consistent plant depicted in the bottom part of Figure 7. Then, to check right consistency, we rename again the right communication events by removing the component ID such that they can be synchronized. Finally, the bottom part of Figure 8 shows a part of the right consistent plant, which is not empty. Now both left and right consistent plants are not empty, but this does not imply the existence of global indeterminate pairs that witness non joint diagnosability. Actually, the part of the left consistent plant depicted here corresponds to two local indeterminate pairs in G_1 with their corresponding left consistent pairs in G_2, i.e., one local indeterminate pair is $((C1.O1.F.O1^*), (O1.C2.O1^*))$ in G_1 with its left consistent pair $((C1.O3.O5^*), (O3.U2.O5^*))$ in G_2 and the other local indeterminate pair is $((O2.U1.C2.F.O1^*), (C1.O2.C2.O1^*))$ in G_1 with its left consistent pair $((O4.C2.O5^*), (O4.C2.O5^*))$ in G_2. While the right consistent plant shown here corresponds to one local indeterminate pair in G_1, which is $((C1.O1.F.O1^*), (O1.C2.O1^*))$, with its corresponding right consistent pair in G_2, i.e., $((O4.C2.O5^*), (O4.C2.O5^*))$. Thus we can see that the same local indeterminate pair, i.e., $((C1.O1.F.O1^*), (O1.C2.O1^*))$ does not correspond to the same consistent pair in G_2 in the left consistent plant and in the right consistent plant, which means that this local indeterminate pair cannot be extended into a global indeterminate pair. In fact, our algorithm gives indeterminate information for joint diagnosable systems that satisfy the following condition: for any set of paths, i.e., one path in the local twin checker of each component and one in the local twin plant for faulty component being a local indeterminate path, that are left consistent and right consistent, respectively, and their corresponding local trajectories in the components cannot constitute an indeterminate pair through synchronization.

Theorem 4: Algorithm 1 has polynomial complexity with the number of system states, exponential complexity with the number of faults and exponential complexity with the number of components.

Proof: From their construction, for a component G_i, the maximum number of states and transitions of the local diagnoser are $(|Q_i| \times 2^{|\Sigma_{i_f}|})$ and $(|Q_i|^2 \times 2^{2|\Sigma_{i_f}|} \times (|\Sigma_{i_o}| + |\Sigma_{i_c}|))$, respectively, where $|Q_i|$ is the number of the component states, $|\Sigma_{i_f}|$ is the number of faults in G_i and $|\Sigma_{i_o}|$ ($|\Sigma_{i_c}|$) is the number of observable events (communication events) in G_i. The maximum number of states and transitions of its local twin plant (local twin checker) are $(|Q_i|^2 \times 2^{2|\Sigma_{i_f}|})$ and $(|Q_i|^4 \times 2^{4|\Sigma_{i_f}|} \times (|\Sigma_{i_o}| + |\Sigma_{i_c}|))$, respectively. In the worst case, the global consistency checking (both left and right ones) consists in synchronizing local twin plant and local twin checkers of all components. Thus, we can conclude that Algorithm 1 has polynomial complexity with the number of system states, exponential complexity with the number of faults and exponential complexity with the number of components. ■

Note that the exponential complexity with the number of faults is for the case where we handle all faults in one component simultaneously. To reduce the complexity, our algorithm is illustrated by dealing with one fault each time.

D. Relaxation of Assumption

In our approach, we have the assumption that all communication events are unobservable, which is a more difficult case than that where communication events are observable. Now we relax this assumption by dividing the set of communication events into two disjoint parts: $\Sigma_{i_c^o}$, the set of observable communication events and $\Sigma_{i_c^u}$, the set of unobservable communication events. In other words, communication events may be observable or unobservable. Algorithm 1 can be reused for this relaxed case after the following modifications.

- For the local component G_F, we construct its local diagnoser by preserving the information about all observable events as well as all communication events, including observable and unobservable ones, and then append to each retained state with fault information, which is the same as that described in Section V-A.
- In this local diagnoser, for each unobservable communication event, we distinguish it between two instances by adding the prefix of L (for left instance, σ being $L : \sigma$) and R (for right instance, σ being $R : \sigma$). Then we reduce the instances as described before and construct local twin plant by synchronizing the two reduced instances based on the set of observable events and the set of observable communication events.
- For other connected normal components, we construct their local twin checker in the same way except that the prefixes should be added only to unobservable communication events in the left and right instances such that the two instances are synchronized based on the set of observable events and observable communication events.
- During left (right) consistent plant construction, unobservable right (left) communication events are distinguished between the local twin plant and local twin checkers by the prefix of component ID. Then the renamed local twin plant and local twin checkers are synchronized based on unobservable left (right) communication events and observable communication events.

VI. Decidable Case

We have proved the undecidability of joint diagnosability when communication events are unobservable. If we assume their observability, then this problem becomes decidable. The reason is that, when all communication events are observable, in the local twin plant and local twin checkers of all components, we obtain all pairs of local trajectories with the same observations, including the same observable communication events. In other words, each path in the local twin plant or local twin checkers corresponds to a pair of local trajectories with the same sequence of communication events. It follows that to check global consistency of local indeterminate paths, the two separate phases for left and right consistency checking becomes only one phase. While in Algorithm 1, the consistency checking separated into two different phases is the reason why it gives only a sufficient but not necessary condition for joint diagnosability. Actually, the observability of communication events makes joint diagnosability equivalent to classical diagnosability since only one phase of global consistency checking implies the same global occurrence order of observations for global indeterminate pairs.

Algorithm 2 presents the procedure to check joint diagnosability when communication events are assumed to be observable. Taking the system model and the fault occurring in the faulty component as input, the parameter, i.e., the current subsystem G_S, is initialized as empty. Then the algorithm begins with the construction of the local twin plant of the faulty component G_F, the current subsystem becoming G_F (line 3-4). Here we emphasize that, due to the observability of communication events, the local twin plant should be constructed by synchronizing two reduced instances based on the set of observable events and the set of communication events, i.e., here communication events do not need to be distinguished by adding prefixes. As long as both current local twin plant T_f and $DirectCC(G, G_S)$ are not empty (line 5), the following steps are repeatedly performed:

- Select one component G_i directly connected to the current subsystem G_S and then construct its local twin checker C_i, which is obtained by first operating delay closure with respect to the set of communication events and observable events and then by synchronizing the two instances based on all events, i.e., the set of communication events and observable events. (line 6-7)
- Synchronize the current local twin plant T_f and C_i based on the common communication events of G_S and G_i, where the newly synchronized FSM is also called the local twin plant for the extended subsystem. (line 8)
- Update the current subsystem by adding this selected component and keep only the paths in the newly obtained FSM that contain ambiguous state cycles with observations for all involved components. (line 9-10)

During this procedure, if the local twin plant T_f for the current subsystem happens to be empty, which means that there is no path that contains ambiguous state cycle with observations for all concerned components. In this case, there is no local indeterminate path that is globally consistent and the algorithm returns joint diagnosability information (line 12-13). Otherwise, if at the end the final FSM is not empty, it is returned by the algorithm as non joint diagnosability information (line 14-15), where any path in it corresponds to a globally consistent local indeterminate path. The reason is that if the communication events are observable, then any path in the local twin plant or a local twin checker corresponds to a pair of local trajectories with the same observations, including the same communication events. So with the assumption of observability of communication events, joint diagnosability checking becomes decidable.

VII. Comparison and Discussion

In this section, we compare joint diagnosability for self-observed distributed systems with classical diagnosability for distributed systems where observable events can be globally observed. In other words, in the latter one, diagnosability analysis requires the global occurrence order of observable events, which is not the case for the former one. Before this

Algorithm 2 Algorithm for checking joint diagnosability with observable communication events

```
1:  INPUT:
    the system model $G = (G_1, ..., G_n)$;
    the fault $F$ and the faulty component $G_F$
2:  Initializations:
    $G_S \leftarrow \emptyset$ (subsystem considered for current checking)
3:  $T_f \leftarrow ConstructLTP(G_F)$
4:  $G_S \leftarrow G_F$
5:  while $T_f \neq \emptyset$ and $DirectCC(G, G_S) \neq \emptyset$ do
6:      $G_i \leftarrow SelectDirectCC(G, G_S)$
7:      $C_i \leftarrow ConstructLTC(G_i)$
8:      $T_f \leftarrow T_f \| C_i$
9:      $G_S \leftarrow Add(G_S, G_i)$
10:     $T_f \leftarrow RetainConsisPaths(T_f)$
11: end while
12: if $T_f = \emptyset$ then
13:     return "$F$ is jointly diagnosable in $G$"
14: else
15:     return $T_f$
16: end if
```

comparison, we briefly recall classical diagnosability definition as follows [20].

Definition 10: (Diagnosability). A fault F is diagnosable in a system G iff

$$\forall s^F \in L(G), \exists k \in N, \forall t \in L(G)/s^F, |t| \geq k \Rightarrow$$
$$(\forall p \in L(G),\ P(p) = P(s^F.t) \Rightarrow F \in p).$$

The above definition states that if F is diagnosable, then for each trajectory ending with the fault s^F in G, for each t that is an extension of s^F in G with sufficient events, every trajectory p in G that is observation equivalent to $s^F.t$ should contain in it F. Informally speaking, the existence of two indistinguishable behaviors, i.e., holding the same enough observations with exactly one of them containing the given fault F, violates diagnosability property. The diagnosability analysis approaches check the existence of such indistinguishable behaviors. With similar idea to indeterminate pairs for joint diagnosability, we call a pair of trajectories p and p' satisfying the following conditions as a **critical pair**:

- p contains F and p' does not;
- p has arbitrarily long observations after the occurrence of F;
- $P(p) = P(p')$.

The existence of critical pairs violates Definition 10 and thus witnesses non-diagnosability.

We can prove that joint diagnosability is stronger than diagnosability for systems with global observations.

Lemma 3: Given two systems G composed of $G_1, ...G_n$ and G' composed of $G'_1, ...G'_n$ such that for each $i \in \{1, ..., n\}$, components G_i and G'_i have the same structure except that all observable events in the component G_i can only be observed by G_i while each observable event in the component G'_i can be observed by all components of G'. In other words, G is a self-observed distributed system and G' is the one with global observations. Then we have the following result: if the fault F is jointly diagnosable in G, then it is diagnosable in G'.

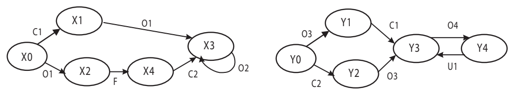


Fig. 9: A simple system model with two components: G_1(left) and G_2(right).

Proof:
Suppose that the fault F is jointly diagnosable in G and that F is not diagnosable in G'. From the non diagnosability of F in G' and G' is a system with global observations, we know that there exists at least one global critical pair of trajectories $p1'$ and $p2'$ in G', i.e., $p1'$ and $p2'$ satisfying three conditions: 1) only one of them contains F, suppose $p1'$; 2) $p1'$ has enough observations after the occurrence of F in all components; 3) $P(p1') = P(p2')$, the projection of $p1'$ to observable events of G' is the same as that of $p2'$, which means that they have the same observations from a global point of view. Now in the self-observed system G, let $p1$ and $p2$ denote the corresponding trajectories of $p1'$ and $p2'$. If we do not consider the difference that each observable event in $p1$ and $p2$ can only be observed by its own component while each observable event in $p1'$ and $p2'$ can be observed by all components, then we have $p1 = p1'$ and $p2 = p2'$. It follows that the fault F is contained in $p1$ but not in $p2$ and after the occurrence of F, $p1$ has enough local observations in each component. Furthermore, note that $p1$ and $p2$ have the same observations, then from the fact that there is no intersection of observable events between components, we can deduce that $\forall k \in \{1, ..., n\}, P_k(p1) = P_k(p2)$. Clearly, $p1$ and $p2$ are an indeterminate pair and thus F is not jointly diagnosable in G, which contradicts the assumption that F is jointly diagnosable in G.

■

If the fault F is diagnosable in G', it is not necessarily jointly diagnosable in G. Actually, if F is diagnosable in G', this means that there is no critical pair in G'. However, this does not imply that there is no indeterminate pair in G. Suppose that there is an indeterminate pair $p1$ and $p2$ in G with $p1$ containing the fault. Then we have $\forall k \in \{1, ..., n\}, P_k(p1) = P_k(p2)$. Now in the system with global observations G', let $p1'$ and $p2'$ denote the corresponding trajectories of $p1$ and $p2$. Then we have $\forall k \in \{1, ..., n\}, P_k(p1') = P_k(p2')$. This does not mean that we can get $P(p1') = P(p2')$, which is however one condition of a critical pair. So there does not necessarily exist a critical pair in G'. In other words, if two trajectories in G have the same enough local observations in all components, their corresponding trajectories in G' may have different observations from global point of view. Thus the existence of indeterminate pairs in G does not imply the existence of critical pairs in G'.

Figure 9 shows a very simple system with two components, G_1 (left) and G_2 (right). Suppose that it is a distributed system with global observations. Intuitively, we can see that for any faulty trajectory, the occurrence of observable event $O1$ is before the occurrence of observable event $O3$. While for any normal trajectory, the occurrence of $O3$ is before that of $O1$. In other words, there is no critical pair and thus the fault is diagnosable in this system. Now suppose that this system is a self-observed distributed system, i.e., the observable events can only be observed by their own component. It is easy to find an indeterminate pair. Consider the pair of trajectories $\rho = (O3, C1, O1, (O2, O4, U1)^*)$ and $\rho\prime = (O1, F, C2, O3, (O2, O4, U1)^*)$. Only $\rho\prime$ is a faulty trajectory and both of them have the same sufficient observations for each component. Thus the fault is not jointly diagnosable. Here we can see that the global occurrence order of observable events makes the fault F diagnosable. While since joint diagnosability does not require global occurrence order of observations, so the fault is not jointly diagnosable.

VIII. Conclusion

In this paper, we consider self-observed distributed systems where observable events can only be observed by their own components. Clearly, the monolithic model of the system is not required, thus the distributed and private (w.r.t. observation) nature of real systems is taken into account. Then we prove the undecidability of checking joint diagnosability when communication events are unobservable, before proposing an algorithm to test a sufficient condition. To check whether there exist indeterminate pairs in the system, we start from local indeterminate paths in the local twin plant and then we check both in sequence left consistency and right consistency. Note that, due to the observation-privacy, the global occurrence order of observable events between different components is not known, which is taken into account through constructing left and right consistent plants separately. At the opposite, in the approaches for DES with globally observable events, twin plant is constructed by incrementally synchronizing local twin plants via both left and right communication events at the same time, which means that in their case, knowledge of the global occurrence order is required. Similar to the distributed algorithms for classical diagnosability ([23], [25], [29], etc.), our algorithm has to construct some part of a global structure, but much less than that in the centralized approach (in particular the components not involved in the algorithm have their model completely not disclosed) and this is normally unavoidable for off-line diagnosability analysis. Afterwards, we adapt this sufficient algorithm for a more relaxed case, i.e., communication events could be unobservable and observable. Then we discuss the decidable case where communication events are observable by giving a formal algorithm to check joint diagnosability. However, the decidable case in [26] is not the same as that in this paper. The former also deals with unobservable communication events, which becomes decidable because of the assumption of exhaustive enumeration about local paths. While here the decidable case is thanks for the observability of communication events. Finally, we prove that joint diagnosability of self-observed systems is stronger than classical diagnosability of globally observed systems with an illustrated example. We see that there is a gap between the decidable and undecidable cases. Next interesting work is to investigate where is the frontier between these two cases, i.e., to study the decidability of joint diagnosability for partial observability of communication events. Another future work is to study the extension of our approach to deal with predictability [30], i.e., the property of the system to be able to predict the fault with certainty before its occurrence considering that in some critical cases, it is very expensive to recover the system after fault occurrence.

Acknowledgment

This work has been supported by the STIC-AmSud DATE project, which is funded by the regional program STIC-AmSud (No.13STIC-04).

References

[1] L. Ye and P. Dague, *Diagnosability analysis for self-observed distributed discrete event systems*, Proceedings of the 4th International Conference on Advances in System Testing and Validation Lifecycle (VALID-12), 2012, pp. 93-98.

[2] S. Tripakis, *Undecidable problems of decentralized observation and control on regular languages*, Information Processing Letters, vol. 9, no. 1, 2004, pp. 21-28.

[3] L. Ye and P. Dague, *New results for joint diagnosability of self-observed distributed discrete event systems*, Proceedings of the 23rd International Workshop on Principles of Diagnosis (DX-12), 2012.

[4] M. Bayoudh, L. Travé-Massuyès, and X. Olive, *Hybrid systems diagnosability by abstracting faulty continuous dynamics*, Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06), Burgos, Spain, 2006.

[5] M. Bayoudh, L. Travé-Massuyès, and X. Olive, *Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis*, Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), Patras, Greece, 2008, pp. 219-223.

[6] M.-O. Cordier, L. Travé-Massuyès, and X. Pucel, *Comparing diagnosability in continuous and discrete-event systems*, Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06), 2006, pp. 55-60.

[7] M. Daigle, X. Koutsoukos, and G. Biswas, *An event-based approach to hybrid systems diagnosability*, Proceedings of the 19th International Workshop on Principles of Diagnosis (DX-08), Blue Mountains, Australia, 2008.

[8] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith, *Distributed state reconstruction for discrete event systems*, proceedings of the 38th IEEE Control and Decision Conference (CDC-00), Sydney, 2000, pp. 2252-2257.

[9] R. Debouk, S. Lafortune, and D. Teneketzis, *Coordinated decentralized protocols for failure diagnosis of discrete event systems*, Journal of Discrete Event Dynamical Systems: Theory and Application 10 (1-2), 2000, pp. 33-86.

[10] R. Debouk, R. Malik, and B. Brandin, *A modular architecture for diagnosis of discrete event systems*, Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02), Las Vegas, NV, USA, 2002, pp. 417-422.

[11] A. Grastien, J. R. Anbulagan, and E. Kelareva, *Diagnosis of discrete-event systems using satisfiability algorithms*, Proceedings of the 22th American National Conference on Artificial Intelligence (AAAI-07), 2007, pp. 305-310.

[12] S. Haar, A. Benveniste, E. Fabre, and C. Jard, *Partial order diagnosability of discrete event systems using petri nets unfoldings*, Proceedings of the 42nd IEEE Conference on Decision and Control (CDC-03), Hawaii, USA, 2003, pp. 3748-3753.

[13] S. Bavishi and E. Chong, *Automated fault diagnosis using a discrete event systems framework*, Proceedings of the 9th IEEE International Symposium on Intelligent Control, 1994 pp. 213-218.

[14] A. Benveniste, E. Fabre, S. Haar, and C. Jard, *Diagnosis of asynchronous discrete event systems, a net unfolding approach*, IEEE Transactions on Automatic Control, 48(5), 2003 pp. 714-727.

[15] C. G. Cassandras and S. Lafortune, *Introduction To Discrete Event Systems*, Second Edition. Springer, New York, 2008.

[16] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, *Supervisory control of discrete-event processes with partial observations*, IEEE Transactions on Automatic Control, 33(3), 1988, pp. 249-260.

[17] O. Contant, S. Lafortune, and D. Teneketzis, *Diagnosis of modular discrete event systems*, Proceedings of the 7th International Workshop on Discrete Event Systems (WODES-04), Reims, France, 2004.

[18] O. Contant, S. Lafortune, and D. Teneketzis, *Failure diagnosis of discrete event systems: The case of intermittent faults*, Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02), 2002, pp. 4006-4011.

[19] E. Fabre, A. Benveniste, and C. Jard, *Distributed diagnosis for large discrete event dynamic systems*, Proceedings of the IFAC world congress, 2002, pp. 237-256.

[20] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, *Diagnosability of discrete event system*, IEEE Transactions on Automatic Control, 40(9), 1995, pp. 1555-1575.

[21] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, *A polynomial time algorithm for testing diagnosability of discrete event systems*, IEEE Transactions on Automatic Control 46(8), 2001, pp. 1318-1321.

[22] A. Cimatti, C. Pecheur, and R. Cavada, *Formal verification of diagnosability via symbolic model checking*, Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, pp. 363-369.

[23] Y. Pencolé, *Diagnosability analysis of distributed discrete event systems*, Proceedings of the 16th European Conference on Articifial Intelligent (ECAI04), 2004, pp. 43-47.

[24] Y. Pencolé, *Assistance for the design of a diagnosable component-based system*, Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI05), 2005, pp. 549-556.

[25] A. Schumann and Y. Pencolé, *Scalable diagnosability checking of event-driven systems*, 20th International Joint Conference on Artificial Intelligence (IJCAI-07), 2007, pp. 575-580.

[26] L. Ye and P. Dague, *Diagnosability analysis of discrete event systems with autonomous components*, Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10), 2010, pp. 105-110.

[27] R. Cori and Y. Métivier, *Recognizable subsets of some partially abelian monoids*, Theoretical Computer Science, vol. 35, 1985, pp. 179-189.

[28] V. Halava and T. Harju, *Undecidability of infinite post correspondence problem for instances of Size 9*, Theoretical Informatics and Applications - ITA, vol. 40, no. 4, 2006, pp. 551-557.

[29] A. Schumann and J. Huang, *A scalable jointree algorithm for diagnosability*, Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08), 2008, pp. 535-540

[30] L. Ye, P. Dague, and F. Nouioua, *Predictability analysis of distributed discrete event systems*, Proceedings of the 52nd IEEE Conference on Decision and Control (CDC-13), 2013, pp. 5009-5015.

Monitoring Virtualized Infrastructure in the Context of Grid Job Execution

Jiří Sitera, Zdeněk Šustr, Boris Parák, and Daniel Kouřil
Grid Department – MetaCentrum
CESNET z. s. p. o.
Zikova 4, Prague, 160 00, Czech Republic
Email: emi-lb@metacentrum.cz

Abstract—**This paper describes a new direction in the development of the Logging and Bookkeeping service, a gLite component tracking job life cycles in high performance computing grids. From its early days, Logging and Bookkeeping is used to track not only jobs themselves, but also the wider details of the job execution environment. Since a large portion of the grid infrastructure is now virtualized, the work at hand concerns tracking the virtualized nature of that runtime environment. With virtualization and cloud technologies being highly flexible and dynamic, the authors believe that it is very important to gather and keep status information for virtual machines used to run the workload. A newly defined monitoring entity – a virtual machine – is integrated with job state information and provides an enhanced view of the current state and history of both the computing job and the underlying infrastructure, as well as their mutual relationship. This paper explains the motivation and discusses the architecture of the newly emerged solution for monitoring virtualized resources – uniquely – in the same context as the workload they are processing.**

Keywords-grid; cloud; virtualization; monitoring; relationship

I. Introduction

This article is an updated and extended version of a work-in-progress report published in September 2012 at the INFOCOMP Conference [1].

Logging and Bookkeeping (LB), part of the gLite grid middleware stack, is a monitoring tool equipped for monitoring the states of all kinds of processes related to grid computing [2]. Besides traditional gLite Workload Management System (WMS) [3] jobs (often refered to simply as "gLite jobs") and logical groupings thereof such as direct oriented graphs (DAGs) or collections, it also monitors input/output data transfers or the states of computing tasks submitted directly to a local resource manager – the CREAM Computing Element (also part of the gLite middleware stack) [4] or TORQUE (Terascale Open-source Resource and QUEue manager) [5].

It collects event information from various grid elements and sums it up to determine the current status of any such process at the given moment. It is designed to accept additional state diagram implementations to support other types of processes as required, relying on essential common features such as reliable event delivery (based either on LB's own legacy messaging layer or standard STOMP/OpenWire messaging), or LB's querying interface. LB is highly security-oriented and has proved itself in WLCG (Worldwide LHC Computing Grid) operations. It is widely deployed across the European Grid Infrastructure.

This article explains how the grid monitoring tool is applied to monitoring the grid's underlying virtualized infrastructure. Section II clarifies what the requirements are and why LB is deemed suitable for monitoring virtualized resources as provided by PaaS (Platform as a Service) clouds. Section III outlines the solution designed and implemented to deliver not only the essential functionality but also to support complex real-world use cases, while Section IV discusses additional issues to consider and focus on in the future. Finally, Section V gives evaluation and results of the work, and Section VI sums up the outcome.

II. Motivation to Include Virtual Machines in the LB Model

Using LB in monitoring virtualized resources is inspired by obvious similarities with the existing processes, backed by explicit requirements from infrastructure operators.

A. Virtual Machine as a Job

LB's main objective is to know everything about the scheduling and execution of computing jobs, not only to respond to "current status" queries, but also to make it possible to analyze the behavior of the infrastructure (failing components, misconfiguration) and possibly even provide certain job provenance capability (ensuring repeatability of jobs/experiments, storing computing environment characteristics and configuration, the description of the compute job provided on submission, etc. – in short, serving as lab notes for "in silico" experiments).

In contemporary grids and other computing infrastructures, machines running computing jobs are themselves dynamic entities following a life cycle similar to that of the grid job itself. It is not unreasonable to expect further blending of cloud and grid models where grid components run either in a cloud (StratusLab [6]), or in a mix with cloud services (MetaCentrum [7], WNoDeS [8]).

All things considered, tracking virtual machines (VMs) throughout their life cycle in contemporary grids is as important as tracking jobs. Moreover, there is an added value to tracking those two kinds of entities in a common manner! Not only does it provide for a better understanding of mutual

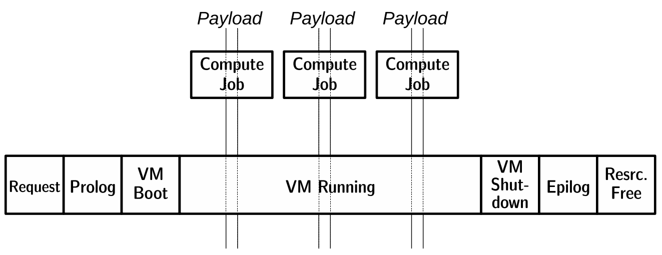


Figure 1. Viewing compute jobs as workload executing over a VM.

relationships and dependencies, but also for a unified view for users and administrators.

Figure 1 shows a simplified and illustrative example of the desired higher-level view of the infrastructure state. It maps compute jobs to the underlying VM life cycle and provides the user with a comprehensive overview of its current state and possible problems. In the case of highly dynamic virtualized infrastructure it can be used to assess efficiency and induced tradeoffs. Data collected in this manner can also be used to produce higher-level statistics and monitoring (mapping actual hardware resources to computing jobs), while the low-level information is still available for detailed inspection if required for debugging or any other kind of ex-post analysis. There are, for instance, scientific projects such as the Grid Observatory, which rely on data provided by LB for behavioral analysis of processes within the grid.

The following LB features are considered most useful in terms of (re)usability for virtual machines:

- Recording primary events and using a state machine specific to the given type of process (job, VM, file transfer, etc.) to combine all information contained therein and determine the current state of that process.
- Providing the ability to get processes grouped or annotated/tagged by the infrastructure, administrators, or users.
- Architecture and implementation based on standards (messaging, authentication and authorization infrastructure, web services), allowing simple event gathering in a reliable and secure way.
- Essential functions (logging events, querying for basic information) provided not only by library functions with bindings for multiple programming languages, but also by command line tools allowing for simple scripting.

There is another key factor – a non-technical one. LB is currently widely deployed across the European Grid Infrastructure. It is not an emerging solution that has yet to be put to production. The support for monitoring underlying infrastructure layers is an evolution of an established and broken in service, making it also a relatively costless solution to the presented problem.

B. Features Requested by the Czech NGI

MetaCentrum, the Czech National Grid Initiative (NGI), is designed as a mixed cloud/grid service, where resources from a single, consistently managed pool can be provided either as traditional batch system-managed resources or as VMs, depending on current user needs [9]. The scheduler (TORQUE) can handle three types of requests:

1) Run a job
2) Run a job in a selected VM image
3) Run a VM

What follows is a summary of feature requests made by the NGI before work presented in this article started. How they were addressed is explained in Section III "Designs and Implementation".

- The desired functionality should provide a uniform, consistent view of the infrastructure, mapping all user requests to actual hardware.
- It should replace currently used data mining tools providing status feeds to the MetaCentrum portal and to the long-term usage statistics processor.
- Since MetaCentrum is also involved in research of batch system scheduling strategies, gathering data relevant to this kind of assessment is another requirement.
- Yet another requirement, albeit one that is already fulfilled by LB's design, calls for an ability to aggregate information from diverse sources (scheduler, virtualization hypervisor, accounting) and even manually triggered state transitions (for instance putting resources in, and taking them out of maintenance).

C. Related Work

1) Regular Infrastructure Monitoring Tools: Tools such as *Nagios/Icinga* or *Ganglia* focus primarily on the "running" state of the given process. Unlike them, this work is not intended to monitor infrastructure health and react to problems, but focuses primarily on understanding the current state of the workload and reporting to users.

Admittedly there is a minor overlap because even LB can be used to provide certain details of infrastructure health. It is namely the LB statistics feature (job/VM status statistics), which, the authors believe, will be further improved by understanding the relationship between the workload and VM layers.

It is, however, important to note that using the above-mentioned monitoring tools to monitor highly volatile short-lived VM instances set up on demand is on the edge of practicality since – looking for instance at *Nagios*, albeit equipped with a selection of plugins – it involves non-trivial, almost continuous reconfiguration of the service not only with machine details but also with access control data. Integrating machine status info with up-to-date references to workload, or vice-versa, would be another challenge, requiring solutions to be designed and implemented on both ends.

2) FSM Implementations: The idea to use a Finite State Machine (FSM) [10], [11] for monitoring purposes is not new. Its typical usage is to monitor software or behavior of network protocols via FSM designed to detect incorrect states or excessive/wrong timing of states. The LB concept is close to existing works describing the application of FSM to monitoring networks of connected components [12], but it is not the same, mostly because LB focuses primarily on workload status, and the behavior of components is a secondary objective.

3) Public Cloud Infrastructures: A typical example of a public cloud infrastructure, the Amazon Web Services (AWS), deploys a complex monitoring tool called Amazon CloudWatch. It is an important brick in the AWS service portfolio [13], allowing not only for monitoring, but also for performing actions (the auto scaling feature). The CloudWatch architecture is based on a central metrics repository fed by resource monitoring tools and providing a common standard API to clients. It also supports user-defined metrics with custom data feeds. The client API has an ability to send notifications and trigger actions when certain thresholds are reached. This service is in many points similar to our proposed solution, but does not employ an FSM.

4) Status Reporting Tools in Scientific Cloud Infrastructures: Each grid infrastructure or cloud management tool has its own way (command line interface, portal) of providing users with the current job or VM status. It is typically implemented as a function of a computing element in a grid or cloud manager (like the Virtual Machine Manager in the case of OpenNebula). Indeed, the LB service is originally one of those tools, monitoring and reporting the status of computing jobs in gLite-based grids.

There is also standardization work in the area of cloud computing, such as the OCCI standard [14] and its implementations, aimed at enabling standard interfaces for reading information off different cloud managers (and even controlling them). However, they do not deal in any way with information from within the virtual appliances, losing a potentially valuable source of information; what is more, they are in no way applicable at the level of workload (grid job) monitoring, and neither is there a separate activity addressing the area of workload status monitoring in a similar way.

To sum up, the authors are not aware of any other work addressing the crucial point – combining available information from different infrastructure layers into a uniform, higher-level, workload-centric view.

III. Designs and Implementation

The proposed solution has been implemented in progressive steps, starting with a pilot implementation on MetaCentrum's OpenNebula instance, running and keeping track of VMs and scheduled TORQUE jobs at the same time. OpenNebula was chosen for the pilot implementation only because it was the cloud manager of choice for the Czech NGI, and the implementation team already possessed adequate expertise for its instrumentation. Apart from that, the solution was in no way tailored specifically to OpenNebula.

This work – i.e., the pilot implementation – was presented and demonstrated at the EGI Technical Forum 2012 [15] and forms the basis of the solution described below.

Next, the work focused on implementing a production-grade solution for MetaCentrum operations (among others

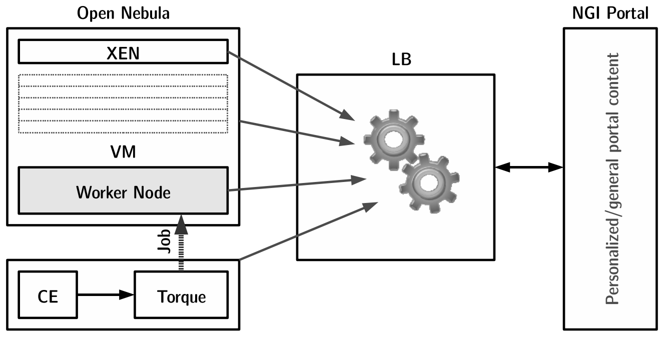


Figure 2. Architecture and components.

instrumenting MetaCentrum's in-house VM manager – Magrathea) and support for federated cloud environments, bringing in additional sources of information external to the batch system and the virtualization stack (administrative operations, information system).

A. Architecture

In the basic architecture used for the pilot, VM life cycle was controlled by the OpenNebula cloud computing toolkit, managed manually by administrators, while jobs were assigned to VMs by a standard grid computing element through an instance of TORQUE. All job-related functionality was already in place (LB-aware TORQUE, [16]). The following sources of events were used to govern the VM life cycle:

- *OpenNebula* – providing hooks for call-out scripts activated on any relevant state change (described in more detail in Subsection III-E)
- *Hypervisor, (specifically Xen)* – generating events showing the current VM state and parameters at hypervisor level
- *Hosted worker nodes* – Operating System running the Worker Node was instrumented (init scripts) to provide independent information from the running VM

LB plays the key role in this concept by combining events from all the above components into one higher-level view. It makes the system more precise and robust, which has been well tested in the context of gLite job monitoring. Obviously on certain occasions, all three sources generate almost identical, i.e., seemingly redundant events. But there is undisputed value even in receiving almost identical events multiple times. It improves reliability, and the comparison between the three events provides for fine-grained job status tracking and simplifies troubleshooting. Besides that, different sources often provide values for different attributes, which are unknown to the others.

Basic system architecture is shown in Figure 2. In that design, the only new feature that had to be implemented was VM instance support in LB, comprising of the VM job type definition, introduction of VM-specific events, and a VM state machine.

Only basic attributes are defined in the VM type to cover the expected virtual machine properties, such as owner identification, memory sizes, or the network status of the VM, detailing the host/domain name, and the type of network connectivity (VLAN, private vs. public). Aside of attributes carried by the VM instance, there is also a solution to keep record of its relationship to other entities actively tracked by LB. That could not be implemented as a simple data structure attribute, and the solution devised for that purpose is discussed in greater depth in Subsection III-C.

The existing set of VM attributes, however, is not necessarily final. LB allows any kind of additional attribute to be simply stored with the instance's status (functionality referred to as "User Tags") with only slight limitations. One cannot, for instance, use relations such as "greater than" or "lower than" when querying for instances by User Tag, since LB does not know the type of that attribute and cannot decide. The only comparison supported is string (in)equivalence.

Each VM instance is identified by a string constructed in the same manner as Job IDs currently used in LB, consisting of the LB server's identification, a short literal denoting the process type, and a random unique string. The randomized unique part of the identifier can be supplied by the registering component if required. In that case, it will only be checked by LB for uniqueness, then accepted. While domain names are not suitable for use as identifiers since they are often recycled (reused by another instance) or even used in alteration by multiple interchangeable VM instances, internal VM identifiers used for instance by OpenNebula (or any other virtualization stack for that matter) are unique to their given OpenNebula server, and can be easily used in the compound ID with the added benefit that one can tell the responsible LB server, the virtualization server and the internal identifier, all at a single glance. Thus, instead of using an LB-generated random ID such as

```
https://lb.example.com:9000/
VM:1ch5-QIGMd_xW3oGM-HScg
```

one may choose to supply their own ID with, for instance, the following format, which is much more informative:

```
https://lb.example.com:9000/
VM:nebula1.example.com_12345
```

B. VM State Machine

The VM state machine is shown in Figure 3. It is based on OpenNebula's internal states but modified to be general enough to provide a single, common view for all VM management systems that will be supported in the future. The states describe major changes in the VM life cycle whereas attributes are used to describe the instance's properties in the current state, or even to distinguish "substates."

It is worth stressing at this point that although the proposed state diagram is considered adequate and detailed enough for the intended purpose, it is relatively easy to implement changes, should it prove necessary. These may range from simple changes in state transitions to extending the state diagram with a completely new state or splitting a single state into two or more. Even with real-world operating experience,

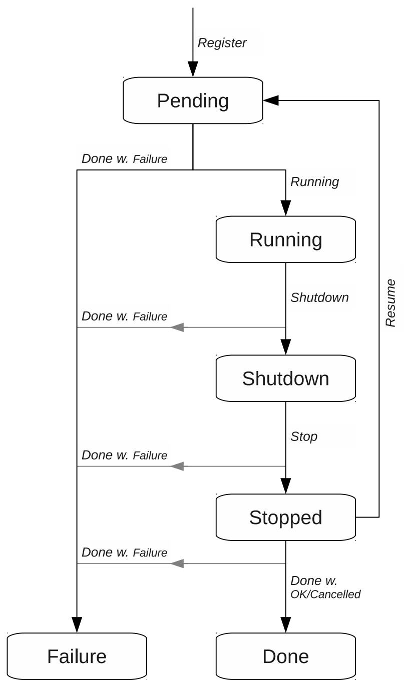


Figure 3. VM state machine.

however, just a single minor change in the mapping of VM states to the generic state diagram was required so far.

Any event received by LB may or may not trigger a change in the state and/or attribute values of an instance. Indeed, some types of events never even attempt to trigger a state change and are only used to bring in new or updated attribute values. (As such, they are not even shown in the state diagram in Figure 3).

As soon as all events received by the current point in time are correctly interpreted, which is an action performed automatically on the arrival of each event, the instance's current state and attributes constitute the most up-to-date set of information as collected from all the various sources mentioned above. LB is designed to overcome obstacles such as events delivered out of sequence, intermediate events not delivered at all, or events received from different sources with clocks skewed in different directions. This is achieved by making the event sorting algorithm rely on arbitrary hierarchical message sequence codes rather than time stamps.

Table I shows how messages from each component/layer (Hypervisor, Cloud Manager, VM Image) contribute to the overall picture of the instance's current state. Since they are all instrumented to produce genuine LB events, the format of

TABLE I. DIVERSE SOURCES OF VM-RELATED EVENTS & ATTRIBUTES

Component	Events	Attributes
CloudManager	Register, Create, Running, Host detail, Shutdown, Done	Hostname, Physical host name, Owner, Requirements, etc.
VM Image	Really Running, Shutdown	Runtime info, user tags
Hypervisor	Running, Shutdown	Actually assigned resources (CPU, RAM, network)

incoming data is unified right from the start of the delivery chain and the LB server can process incoming data uniformly regardless of the source component.

To allow universal queries to the LB server, VM states (as well as states defined in other state diagrams implemented in LB) are also mapped to states in the default LB state machine used primarily for gLite WMS jobs. Thanks to that mapping, users can easily query LB for, e.g., all their running tasks regardless of whether they are computing jobs, virtual machines, or any other kind of supported process. The complete mapping between VM and gLite job states is shown in Table II.

TABLE II. MAPPING OF VM STATES TO GENERIC GLITE STATES FOR UNIVERSAL QUERIES

VM State	Generic State
Pending	*Submitted* for freshly registered VMs
	Waiting for VMs resumed from *Stopped* state
Running	*Running*
Shutdown	*Waiting*
Stopped	
Done	*Done*, distinguished by the *done_code* attribute
Failure	

C. Relationship of Entities

Limited support for specific inter-job relationships was inherited from previous LB versions. Relationships between different types of jobs were implemented differently, depending on then-existing requirements. Experience with that implementation was taken into account when designing the new solution.

1) Pre-existing Implementations: Only the following job type pairings were supported before the generic solution was designed:

- *Parents/children* in DAGs (Direct Acyclic Graphs) and job collections, where one single parent job is linked to its children and vice versa. This is implemented by an extra database attribute in all children, which explicitly states the ID of the parent. This attribute can be used directly in the LB server's database queries and, being also implicitly indexed, allows for a quick reconstruction of the whole set of children in a collection.
  The relationship is established on registration, wherein status records are created for the parent job as well as all the children in a single server-side step, filling in the parent reference. There is no support for removing jobs from collections, hence the relationship can be never canceled. Neither is there currently support for growing collections already registered.
- *Compute jobs/Sandbox transfers*, where each gLite compute job can maintain a reference to its input and output sandbox transfers. There are specific single-purpose attributes included in the job's internal status structure, one for the input, another one for the output data. They will contain job IDs of single sandbox transfers, or collections thereof. Since the sandbox transfer IDs are encoded in the internal status of the job, they are slow to access unless the LB administrator has set up specific database indices for that purpose. This makes searching for compute jobs by input or output sandbox transfer ID possible but impractical.
  The relationship is established by a special-purpose *sandbox* event that will set or modify the reference, but obviously the job status structure can never refer to multiple input or output sandbox transfer IDs at once. Collections must be used if multiple input or output sandbox transfers are to be covered.

Both features remain, for now, available alongside the generic solution outlined below. While the latter could be adequately replaced with the new approach in the future, the former presents a very specific case with built-in server-side logic (registration, state and histogram algorithms, etc.), and the functionality must remain unchanged.

2) Designing a Generic Bilateral Relationship Record: Facing the task to implement yet another type of relationship – that between a virtual machine and its workload – it was decided to take a generic and more widely applicable approach that could support any kind of bilateral inter-job relationship in a common manner once and for all. The requirements were as follows:

- Support $m : n$ relationships as a VM will definitely relate to multiple compute jobs as well as a compute job can relate to multiple machines (typically in cases where it did not succeed in one and had to be resubmitted to a different resource).
- Allow distinguishing between active and past relationships.
- Avoid adding complexity in server-side processing, i.e., avoid, wherever possible, dependence on additional database queries when registering relationships.
- Support establishing or canceling the relationship at any time during the participating entities' lifetimes.
- Since it was decided to make this a generic functionality, allow for specifying the nature of the relationship, i.e., what does the relationship record mean.

To that end, a new type of event (*relationship*) has been introduced to control the relationship records, and the present SQL schema was extended with an additional table comprising of the following attributes:

- *Source job*: the job the relationship originates from.
- *Target job*: the other job in the relationship. Note that the source/target approach obviously makes the record asymmetric. To achieve symmetry, every *relationship* event actually results in two separate records being created, one of them having the originating job as the source in one instance, and as a target in the other instance.
- *Target job type*. It is assumed that relationship information is never retrieved from the LB server separately, but always together with the status of the

"source" job, which also indicates its type. Therefore, the type of the "source" job does not have to be stored with the relationship record and still the types of both jobs in the relationship are known and can be used to interpret the relationship's meaning. For instance, the existence of a relationship between a compute job and a virtual machine can be easily interpreted in the sense that the job is running on that virtual machine. A relationship between a virtual machine and a sandbox transfer, e.g., will be interpreted in the sense that LB was monitoring the transfer of the virtual machine's image during the prolog stage.
Technically, a relationship can be registered for any exotic combination of two job types. Sometimes the meaning may be obvious, as in the two cases above, and sometimes not. At any rate, interpretation is left to the party who has logged the relationship in the first place.
To minimize server-side overhead, the client is required to specify the *target job type* along with its ID when logging a relationship. This allows the server to register the relationship without having to unparse the other job's status – a relatively costly operation. There is a minor danger that the client will log a wrong job type, but it is felt that keeping the data correct and reliable is always the client's responsibility. Still, implementing an option to read the target job's type from the database regardless of the extra cost is an opportunity for future development.

- *Relationship status*. The status is given as an enumerated type, allowing for future extensions. The initial implementation recognizes the following states:
  - *Active*: in the context of virtual machine/workload monitoring, an *Active* relationship will refer to the job currently running on the machine.
  - *Inactive*: primarily for relationships that are not active anymore. Possibly also for relationships that will become active in the future (for instance a job has been scheduled to a particular resource but has not arrived there yet). Once again, the precise interpretation of the meaning may be up to the party who has logged the relationship.
  - *Canceled*: used instead of removing the relationship record. The physical relationship record is only removed when one of the relevant jobs is being purged from the LB server for good.

Note that a relationship record carries only information valid at present and does not maintain any record of the relationship's history. However, since relationships are established and controlled through events, their history can always be reconstructed by querying for the logging history (i.e., raw events) of jobs participating in the relationship.

D. Security Considerations

The security of data gathered by LB is an important part of the solution. LB implements a messaging infrastructure to deliver events from the place where they are created to the LB server. The infrastructure is built upon a set of mediators (*inter-loggers*) that provide a secure and fault-tolerant transport of messages between LB clients and servers. This also means that the actual worker nodes do not necessarily require direct Internet access since they are sending their events through the interloggers, typically installed only on head nodes with different network accessibility settings.

All connections within the infrastructure are authenticated, including the delivery of a new event. Upon receiving a connection carrying an event, the LB server makes an authorization decision to see if the originating component is entitled to log that kind of event for the given job.

Reading access is likewise subject to similar security precautions. Every client querying the LB server must be properly authenticated and authorized to obtain the data requested. There are actually two ways of setting up access: both are applicable to all processes monitored by LB, including virtual machines:

- *Server-wide authorization policy*: there are several authorization categories to grant rights across all jobs-processes known to the server. The categories cover both the logging and querying parts of data processing. There is a specific category, for instance, to allow logging events specific to workload managers. Only grid components listed in that category can log those specific kinds of events. Similar categories are there for different levels of reading access.
- *Per-job ACLs*: an Access Control List can be maintained for any individual entity (job, VM, etc.), granting additional permissions to read or log events for that entity alone. The ACL is stored as a part of the entity's internal state, maintaining a list of allowed or banned users. Users are identified by DN or Kerberos principal, which can be used interchangeably, and mapped one to another by means of a server-side Globus-compliant *gridmap* file [17].
  Unlike the server-wide policies, ACLs on a job are entirely under the control of the owner of that job. Users can therefore specify fine-grained access control for their jobs.

Per-job ACLs are used in virtual machine monitoring to overcome the fact that OpenNebula currently cannot communicate the assigned ID to the virtual machine it has instantiated. When creating a new virtual machine, the ACL of the corresponding instance in LB is populated with its identity. Using appropriate credentials later on, the virtual machine can look up its appropriate job ID when needed.

In order to simplify the deployment of LB in diverse environments, LB supports multiple security mechanisms. In particular, the de-facto grid standard using X.509 and TLS/SSL has been supported from the very beginning. Support for Kerberos has been added recently. It is, however, implemented in a generic way, so that adding additional security mechanisms is quite feasible.

The LB server also supports mapping of client identities by means of a *gridmap* file so that a single person relying on different clients using different types of credentials obtained through different authentication mechanisms can always be identified as the same user. This "mixed" setup where users use their Kerberos tickets for regular work in the command line, and their X.509 certificates to access LB over the HTTPs interface, is actually used in MetaCentrum where LB was first deployed to monitor the virtualized infrastructure along with the computing jobs.

E. OpenNebula Instrumentation

OpenNebula implements a system of so-called Virtual Machine Hooks. Hooks are programs automatically executed, or *triggered*, when a virtual machine changes its state. This allows us to provide simple and standard modular implementation of OpenNebula instrumentation for LB. An LB hook for OpenNebula has been implemented as a stand-alone script invoked by hooks using supplied configuration files [18].

OpenNebula supports hooks for all the relevant states, shown in Table III. Other hooks are available, ensuring future extensibility of this solution.

Information about the virtual machine is passed to the script in the form of an XML template. Each state change triggers a hook, passing unique virtual machine identifier and its template as arguments to the registered executable.

The instrumentation script is written in the Ruby programming language, which is the language of choice for OpenNebula's modules and extensions. It should be registered for all the relevant virtual machine states mentioned above. The script requires a Base64-encoded virtual machine template and the name of the hook as arguments. Other optional arguments include logging method, log file location for debugging purposes or map file location for dynamic mapping of user identities.

Authentication of the instrumentation script as a trusted LB events source can be performed using both X.509 or Kerberos-based host credentials.

The script itself performs four basic actions:

1) Decodes and parses the virtual machine template.
2) Maps user identities (e.g., user names to their X.509 certificate DN).
3) Performs data transformation: currently computes the overall VM runtime from runtime values on individual hosts.
4) Constructs and sends an event to LB using its native API and local binaries

Events are constructed using ERB, Ruby's templating system. The templates are easily modifiable without an extensive knowledge of the programming language itself.

TABLE III. OPENNEBULA STATES WITH HOOKS

State	Trigger Event
Create	Virtual machine has been submitted by the user
Prolog	OpenNebula's scheduler found an appropriate host and started deployment
Running	Virtual machine is running and ready to accept jobs
Shutdown	Virtual machine is shutting down
Stop	Virtual machine has been stopped (temporarily)
Done	Final state, end of the virtual machine life cycle
Failed	Previously issued action has failed, virtual machine is not available

F. Support for Complex Use Cases

The newly developed solution is not limited to basic interaction with a single instance of OpenNebula, but rather implements all features required for other, more complex scenarios.

1) MetaCentrum: Providing a unified view of resources and job execution for users and administrators, this use case follows requirements described in Section II-B with the addition of LB being used as a part of a distributed implementation of TORQUE, recently deployed in MetaCentrum [16]. Here, LB acts as a service providing users with job status information via a modified *qstat* utility.

MetaCentrum operates its own cloud manager called Magrathea [19]. Its role is to cooperate with the TORQUE batch manager to provide integrated grid and cloud environment running within a single resource pool. To use LB to combine status information from both kinds of workload – virtual machines and batch jobs alike – into one overall picture, support for LB had to be implemented in Magrateha. The instrumentation of Magrathea to send LB events in a manner similar to the OpenNebula case was straightforward. The common VM state diagram was tuned to work correctly with both event sources (OpenNebula and Magrathea) at the same time. Now each MetaCentrum user can use all three worlds (OpenNebula, Magrathea, batch jobs) and see all the respective status information in one common LB service.

2) FedCloud: The FedCloud Task in EGI (European Grid Infrastructure) deals with federation of resources in a cloud environment. Here, the aim is to achieve interoperability of different cloud solutions, running in different administrative domains and on different cloud manager implementations. The role of LB in providing global status info for the whole federated infrastructure is similar to that, which LB plays in gLite-based grids such as WLCG – i.e., monitoring all processes across the infrastructure, regardless of geographical site, ownership or flavor of the underlying technology, as long as it is instrumented to deliver events to LB.

This particular use case requires LB to support multiple hypervisors and cloud managers (equivalent to supporting different job managers – computing element implementations – in grid job monitoring). There is currently support for OpenNebula with XEN, and the work on supporting OpenStack and KVM is underway. It is important to stress that the implementation of such additional support consists solely in client-side work, i.e., in instrumenting the new sources to generate LB events, or possibly finding ways to translate existing streams of outgoing information into LB events. The LB service as such is already prepared for the task.

This use case is also going to rely on the multitude of ways one can access LB from the user perspective. LB can be queried through different interfaces, or configured to become a producer of messages delivered over different channels. Some of the interfaces can be accessed with generic, widely available clients such as Web browsers or RSS readers. Figure 4 shows LB as a message processor capable of receiving a flow of messages, potentially over diverse channels, and making the processed information available either on query, or over another streaming channel.

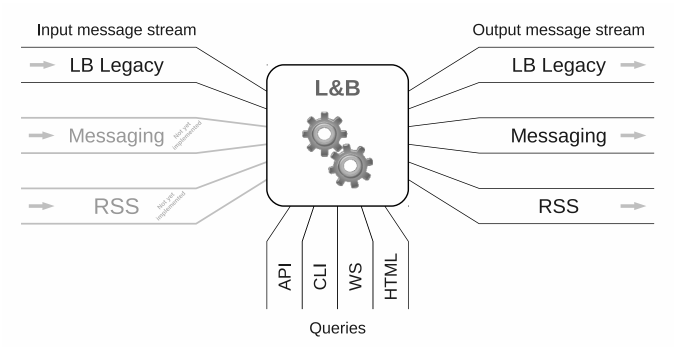


Figure 4. LB as a message processor publishing information over different output channels.

LB has the potential to provide complete infrastructure for common cloud status monitoring, taking care of all the stages from data sources, through data transfer, interpretation and storage, to client access.

3) Support of User Group Workflows: Compared to traditional computing jobs, VMs are a little specific in that they always need to be assigned workload when running (i.e., having started for the first time, recovered from a downtime or finished migration), which makes them actually very similar to pilot jobs! Pilot jobs are simple computing jobs carrying no workload on submission, but rather waiting until they acquire the necessary resources and only then receiving the actual workload, i.e., computing work to be done.

Thus a pilot job framework is a good example of a user-specific workload management system. It is designed to distribute workload to job slots at the moment when pilot jobs (or – analogically – VMs) actually start. It may be very convenient for such a framework to receive notifications of relevant VM status changes, rather than heving to repeatedly poll all relevant resources. That is easily achieved with LB notifications generated on pre-determined conditions and sent out over LB's own legacy messaging chain or through a STOMP or OpenWire-enabled messaging broker.

Users may choose, for instance, to be notified any time any of their machines reaches state *running*. More elaborate sets of conditions are also supported. The resulting notification contains the full VM status information and, if requested on registration, also the full history of events for that machine so far.

Another option to receive updates on VM state changes would be RSS. The RSS interface in LB is as elaborate as the job querying interface, allowing for the creation of highly customized feeds, which can be then received with any RSS reader, provided it can handle X.509 authentication.

G. State Machine for Physical Machines

This is, at the same time, a usage scenario possible with the current implementation, and a consideration for future work.

As per the original design, VM instances use their attributes to refer to their respective physical hosts only by name (the FQDN – Fully qualified domain name, actually) and no track is kept of the actual status of those resources. But there is an obvious similarity between physical and virtual machines. If anything, physical resources are even simpler to describe than virtual ones, and a VM state diagram is easily applicable to physical machines. So the option is to register physical resources as "VM" instances as well, and reference the identifier instead, either by using the ID assigned by LB rather than the FQDN, or by registering a bilateral relationship between the virtual machine and its physical host as described in Subsection III-C.

With that done, the same level of detail can be provided for virtual and physical machines alike, although some supported states can pick up different meanings in the physical world (for instance state *pending* would not mean that the machine is being set up by the virtualization stack, but rather that it is being installed by its administrator) or remain unused altogether. Maintaining detailed status information for physical machines is important because sites need to keep operational logs of physical resources management (maintenance, testing, repair) and understand it correctly in the mixed grid/cloud model (providing proper hardware usage statistics).

Events governing the status of a physical machine record – be it an instance of the VM type used in an "overloaded" mode, or a newly designed Physical Machine type – can be generated:

1) automatically by the machine itself or, more specifically, by its operating system's image, properly instrumented and contextualized:
   - machine start
   - regular machine shutdown
2) automatically by infrastructure monitoring tools:
   - machine down (when unreachable or otherwise recognized as being down)
3) manually by resource administrators
   - machine being installed
   - machine being moved or maintained
   - machine failed (due to a HW issue, for instance)
   - any additional operational records can be logged as *UserTags*

Basically, then, the main distinction lies in the fact that events logged for virtual machines by the Cloud Manager are generated by a human administrator for the physical ones.

There are other benefits stemming from the fact that virtual and physical resources are treated similarly. Figure 1 was showing workload executing over a VM. By including physical resources into the picture, one can also view workload running directly on the physical machine (where allowed by the actual solution – for instance Magrathea or WNoDeS), or watch jobs executing in virtual machines over their physical hosts – see Figure 5.

IV. Future Work

There are several topics identified as a potential improvement or extension of the existing solution. Potentially important as they are, they were already envisioned and accommodated for at design time but the decision on their actual implementation was left for the future.

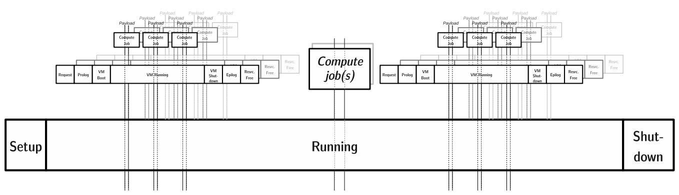


Figure 5. Virtual machines and native jobs executing on a physical machine.

A. *Virtual Cluster Implementation*

The Virtual Cluster service provided by MetaCentrum can create multiple VM instances per request [7]. All the resulting VMs have common attributes (type of network connection) and are closely related. Similar functionality is provided by other node-on-demand services such as WNoDeS [8].

In most cases, all nodes in a virtual cluster set up in this manner are intended to be used together in close conjunction, usually to run parallel computing jobs, which is why a user can benefit from easily obtaining an overall view of the state of the cluster.

It may be a good idea to reuse the "collections" functionality in LB, typically applied to grid jobs or sandbox transfers. From the user's point of view the state of the collection combines the states of all its members. Individual VM details are still accessible under the VM instance's own ID – the collection functionality simply adds another identifier (collection ID) to access aggregate information for the whole collection, such as child (member) status histograms. The fact that LB's VM type can also be used to monitor physical machines as discussed in section III-G also makes such collections applicable to hybrid clusters where a fixed physical cluster is extended (perhaps temporarily) with additional virtual nodes to improve peak computing power.

There are only minor differences between (already supported) job collections and (proposed) VM collections to address, chief among them the understanding of the overall status of such collections. While a computing job collection can be considered running, for instance, as long as at least one of its children is running, a VM collection should not be considered running unless all the VMs in that collection – or at least a certain majority – are up. Other collection-wide states require similar redefinition. But apart from that, all the essential functionality exists and can be applied to the new collection type.

B. *VLAN Status*

Virtual Cluster services offered by MetaCentrum provide not only sets of machines but also networking connections in the form of virtual Ethernet (VLAN) [20], thus offering a comprehensive IaaS (Infrastructure as a Service) solution. The VLANs have their own life cycle managed by a purpose-built VLAN manager (SBF) [21], which could become a source of events for LB. After all, a *network* is recognized as an entity in its own right by many cloud-related standards such as OCCI (Open Cloud Computing Interface).

An ability to track the state of the network together with its attributes (private vs. public network, additional services such as tunnels, NAT (Network Address Translation) or firewalls) could be valuable in many scenarios. True, it would require another state machine to be implementeda, but that has become a relatively routine task recently, and would be well justified by an interesting use case.

C. *State Diagram Evolution*

Although the state diagram explained in Subsection III-B currently seems to meet the intended need, there were suggestions that it should cover additional states used in more elaborate scenarios possible with advanced cloud managers, such as the VM *migrating*, *resizing*, etc. As it is, these states, for instance, are inherently the substates of the *Pending* state, and even with the current implementation they can be distinguished from other such substates by means of user tags, or – with a simple extension of the code – by means of an additional state attribute.

On the other hand, as long as there is a use case to sufficiently justify the need for making these into separate states, implementing an extension to the state diagram is sufficiently straightforward.

V. Results and evaluation

The new experimental version of LB implementing the features described in this paper was evaluated in MetaCentrum pre-production environment for a few months and currently runs in production. All the code is considered well tested, and it is included in the official LB release, starting with version 4.0. On top of that, TORQUE instrumentation code is available with MetaCentrum's TORQUE patches, and OpenNebula instrumentation can be obtained as a set of documented hook scripts [18]. Thus, all tangible results of the work are publically available.

Based on previous work and the new experience with a production LB instance gathering all information about jobs and virtual machines running in MetaCentrum, the following results can be summed up and evaluated:

1) Performance: Since the whole event delivery chain is ansynchronous, the overhead on infrastructure components is negligible. The authors have previously performed extensive measurements, showing that the processing capacity for a real-world spread of jobs amounted to several hundred thousand jobs per day [22], and the declared target of a throughput of 1 million jobs per day was achieved at least for simple jobs.

Table IV compares known performance limits to measured production load extrapolated from eight months of production use in MetaCentrum, a mid-sized grid infrastructure. It shows that a typical combined grid/cloud operation uses up only a small portion of the possible throughput, making the actual overhead negligible.

TABLE IV. Comparing real-world load in the Czech national grid with the maximum capacity

Item	**Prod. per Year**	**Maximum Capacity**
Computing jobs, mixed complexity	1,550,000	$\sim$90,000,000
Virtual machines	25,000	$>$90,000,000
Individual Events	500,000,000	$>$1,550,000,000

2) Applicability of Results: The common monitoring solution succesfully collects and, most importantly, correlates workload and infrastructure status data. Evaluation of the amount and structure of data collected and inferred by LB shows that it indeed provides a detailed, unified view of the multi-layered virtualized environment, and that its output can be used not only to check current status, but also to feed higher-level statistics and reporting systems.

An internal feasibility pre-study performed at the Czech NGI, for instance, shows that its current statistics gathering mechanisms could be replaced with an LB-based solution, as long as LB provides a mechanism to monitor physical machines as discussed in Section III-G, with the added benefit that it wolud completely cover also the emerging PaaS and IaaS cloud services – a task that the existing statistics gathering solution cannot perform.

VI. Conclusion

This paper shows how the potential of an existing job-monitoring infrastructure can be reused in the virtualized world. The design and implementation of LB was extended to support virtual machines as a new kind of monitored entity, and the new functionality was demonstrated in real-word usage. LB can now handle mutual relationships between various entities involved in a modern computing environment (dynamic sets of virtual machines available directly to users, and potentially hosting jobs managed by a grid service). Although this work was, at its beginning, primarily driven by the Czech NGI's requirements, it was found useful at a much wider scope. Typical motivations involve resource federation in the cloud-oriented world. The authors are proposing this solution to FedCloud, the cloud interoperability task force acting within the European Grid Infrastructure. LB, with its established presence in gLite-enabled grid sites across the European Grid Infrastructure, resulting in easy adoption, can be a reasonable candidate for a monitoring and notification service in emerging international "cloud-like" scientific environments.

Acknowledgement

This work is part of the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005).

Fundamental development and maintenance of the Logging and Bookkeeping service was co-funded by the European Commission as part of the EMI project under Grant Agreement INFSO-RI-261611.

References

[1] Z. Šustr and J. Sitera, "Understanding virtualized infrastructure in grid job monitoring" in *INFOCOMP 2012, The Second International Conference on Advanced Communications and Computation*, Venice, Italy, pp. 167 – 170, 2012.

[2] "Logging and bookkeeping," 2008. [Online] Available: http://egee.cesnet.cz/en/JRA1/LB/. [Accessed December 20, 2013].

[3] M. Cecchi et al., "The gLite workload management system," *J. Phys.: Conf. Ser.*, vol. 219, 2010.

[4] P. Andreetto et al., "Status and developments of the CREAM computing element service," *J. Phys.: Conf. Ser.*, vol. 331, 2011.

[5] G. Staples, "TORQUE resource manager," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing (SC)*, 2006.

[6] "StratusLab," 2012. [Online] Available: http://stratuslab.eu/. [Accessed: December 20, 2013].

[7] M. Ruda et al., "Virtual clusters as a new service of MetaCentrum, the Czech NGI," CESNET, 2009. [Online] Available: http://www.cesnet.cz/doc/techzpravy/2009/virtual-clusters-metacentrum/. [Accessed: December 20, 2013].

[8] D. Salomoni et al., "WNoDeS, a tool for integrated grid and cloud access and computing farm virtualization," *J. Phys.: Conf. Ser.* 331 052017, 2011.

[9] J. Sitera, M. Ruda, P. Holub, D. Antoš, and L. Matyska, "MetaCentrum virtualization – use cases," CESNET, 2010. [Online] Available: http://www.cesnet.cz/doc/techzpravy/2010/metacentrum-virtualization-use-cases/. [Accessed: December 20, 2013].

[10] J. E. Savage, "Models of computation: exploring the power of computing," Addison-Wesley Pub, 1998.

[11] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme, "Modeling software with finite state machines: a practical approach," Auerbach Publications, 2006.

[12] B. Birregah, K. H. Adjallah, K. S. Assiamoua, and P. K. Doh, "Grid systems monitoring and assessment using finite state machines with median symmetry operators," in *IEEE International Conference on Systems, Man and Cybernetics*, Montreal, pp. 741 – 746, 2007.

[13] J. Varia, "Architecting for the cloud: best practices," 2011. [Online] Available: http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf. [Accessed: December 20, 2013].

[14] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, "OCCI specification," OCCI-WG OGF, 2011. [Online] Available: http://occi-wg.org/about/specification. [Accessed: December 20, 2013].

[15] Z. Šustr et al., "Monitoring national infrastructure with L&B," in EGI Technical Forum, 2012. [Online] Available: http://youtu.be/tI5m45jbxmU. [Accessed: December 20, 2013].

[16] M. Voců et al, "Using L&B to monitor TORQUE jobs across a national grid," in *EGI Community Forum 2012 Book of Abstracts*, Garching, Germany, 2012.

[17] "Gridmap," 2007. [Online] Available: http://dev.globus.org/wiki/Gridmap. [Accessed: December 20, 2013].

[18] "gLite LB instrumentation Scripts for OpeNebula," 2012, [Online] Available: https://github.com/CESNET/metacloud-lb-scripts. [Accessed: December 20, 2013].

[19] M. Ruda, J. Denemark, and L. Matyska, "Scheduling virtual grids: the Magrathea system," in *VTDC '07 Proceedings of the 2nd international workshop on Virtualization technology in distributed computing,* Article No. 7, ACM, 2007.

[20] D. Antoš, L. Matyska, P. Holub, and J. Sitera, "VirtCloud: virtualising network for grid environments – first experiences," in *The 23rd IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Bradford, UK, 2009.

[21] Z. Šustr et al., "MetaCentrum, the Czech virtualized NGI," in *EGEE Technical Forum 2009*, Barcelona, Spain, 2009. [Online] Available: https://egee.cesnet.cz/cs/info/virtualizace.pdf. [Accessed: December 20, 2013].

[22] Z. Šustr et al., "Mass testing of EMI products in Czech NGI's virtualized environment," in *EGI Community Forum 2012*, Garching, Germany. [Online] Available: http://egee.cesnet.cz/cvsweb/LB/CF12-mass-test.pdf. [Accessed: December 20, 2013].

Static Preprocessing for Automated Structural Testing of Simulink Models

Benjamin Wilmes
Berlin Institute of Technology
Daimler Center for Automotive IT Innovations (DCAITI)
Berlin, Germany
E-Mail: benjamin.wilmes@dcaiti.com

***Abstract*—A feasible automation of testing software models would be of great benefit to industry, given the advantages of early testing as part of an efficient quality assurance process. Despite search-based testing having been applied with promising results to automate structural test data generation for Simulink models, the approach lacks efficiency. This paper features three static-analysis-based preprocessing techniques which are carried out prior to an automated test data search, to mitigate this efficiency problem. The first technique identifies unsatisfiable coverage goals by analyzing the ranges of model internal signals and excludes them from the search. The second preprocessing technique aims at reducing the search space by analyzing which model inputs actually require stimulation in order to reach a certain model state. A third technique sequences the coverage-goal-related search processes in order to maximize collateral coverage and reduce the size of the generated test suite. These additional techniques are able to make the search-based approach considerably more efficient, as results of a case study with our search-based testing tool TASMO, applied to industrial Simulink models, reveal.**

***Keywords*-Search-Based Testing, Static Preprocessing, Automotive Industry, Simulink.**

I. Introduction

First of all, note that this paper is an extended version of a previous publication [1]. It contains further details on the topic, additional illustrations and an extended case study.

In many of today's application areas, the creation of embedded controller software relies on model-based design paradigms. Various industries, such as the automotive industry, use Matlab Simulink (SL) [2] as the standard tool to create and simulate dynamic models along with a code generator, for instance TargetLink (TL) [3], in order to automatically derive software code from such models.

As they are normally the first executable artifacts within software development processes, SL models play an important role in testing theory. Industrial testing practice, however, usually focuses on higher-level development artifacts, like testing integrated software or systems as a whole. This discrepancy has both traditional and practical reasons. On one hand, current testing processes still require further adaptation to the model-based paradigm. On the other hand, companies are pressed for time in product development and must deal with an increasing demand for innovation. This can lead to a disregard for low-level tests and model tests in particular. Yet focusing too one-sided on tests of higher-level software or system artifacts poses the risk that faults may be found late in the process, which can lead to increased costs, that some faults can hardly be discovered on higher levels, or that certain functionality is not tested at all.

Thus, automating the testing of software models is highly desirable in industrial practice, particularly with regard to what is normally the most time-consuming testing activity: the selection of adequate test cases in the form of model input values (test data). Search-based testing is a dynamic approach to automating this task. It transforms the test data finding problem into an optimization problem and utilizes meta-heuristic search techniques like evolutionary algorithms to solve it. Search-based testing [4] has been studied widely in the past and has also been applied successfully for testing industrial-sized software systems [5]. Both structural (white-box) and functional (black-box) testing can be automated with the search-based approach.

Zhan and Clark [6], as well as Windisch [7], applied search-based testing to structural test data generation for SL models. The work of Windisch not only supports Stateflow (SF) diagrams (which are used fairly often in SL models), but also makes use of an advanced signal generation approach in order to generate realistic test data. While his approach has led to promising results in general, outperforming commercial tools in terms of effectiveness, it lacks efficiency when applied to larger models. Furthermore, it shows difficulties targeting Boolean states and tackling complex dependencies within models [8].

The work presented in this paper is a first step toward overcoming some of these shortcomings by exploiting static model analysis techniques before the search process actually starts. These techniques will therefore be referred to as static preprocessing techniques. Our scope is test data generation for TL-compliant Simulink models. Our primary aim is to improve the efficiency of the approach by Windisch. While our work is targeted at SL models, we believe that the presented concepts are generally portable to similar data-flow diagram types and, at least conceptually, even to structural testing of code.

This paper is structured as follows: Section II introduces search-based testing and its application to structural testing of SL models. Section III presents our approach to supporting the search-based technique by integrating three preprocessing techniques. In detail, we present a signal range analysis (Section III-A) which captures range information of internal model signals and, in this way, allows partial detection of unreachable model states. We then propose a signal dependency analysis for the purpose of search space reduction (Section III-B). Our third contribution is a sequencing approach which derives an order in which coverage goals of a structural test are processed by the search (Section III-C). Insight into our tool prototype and a case study are provided in Section IV and V. An overview of related work in the field of structural test data generation for SL models is provided in Section VI, followed by our conclusions in Section VII.

II. Background

A. Search-Based Structural Testing

Initiated in the 1970s by Miller and Spooner [9] and revived by Korel in the 1990s [10], search-based testing [4] and its application to industrial cases has been extensively studied in the last decade.

The general idea of the search-based approach is pretty simple: a test data finding problem (which surely differs in its nature depending on the kind of testing) is transformed into an optimization problem by defining a cost function, called fitness function. This function rates any test data generated by the deployed search algorithm - usually based on information gained from executing the test object with it. The rating must express, in as much detail as possible, how far the test data is from being the desired test data. An iteratively working search algorithm uses these fitness ratings to distinguish good test data from bad, and based on this, generates new test data in each iterative cycle. This fully automated procedure continues until test data satisfying the search goal(s) has been found, that is, if a fitness rating has reached a certain threshold or until a predefined number of algorithm iterations have been performed. Various search algorithms have been used in the past. Due to their strength in handling diverse search spaces, evolutionary algorithms, like genetic algorithms, were often preferred [11].

Applied to functional testing, the search-based approach is generally utilized to search for violations of a requirement. In this case, a sophisticated fitness function needs to be designed manually when following the standard approach. However, when applied to structural test data generation, fitness functions can be derived completely automatically from the inner structure of the program to be tested.

Structural testing is commonly aimed at deriving test data based on the internal structural elements of the test object, e.g., creating a set of test data which executes all statements of a code function, or all paths in the corresponding control flow graph. Industrial standards like ISO 26262 even demand the consideration of coverage metrics when performing low-level tests. Search-based testing can automate this task for various coverage criteria (like branch or condition coverage) by treating each structural element requiring coverage as a separate search goal, called a coverage goal (CG/CGs in plural). Each CG is accompanied by a specific fitness function. Wegener et al. [12] recommend composing the fitness function of the following two metrics: approach level (positive integer value) and branch distance (real value from 0 to 1). Given a test data's execution path in the control flow graph of the test object's code, the approach level describes the smallest number of branch nodes between the structural element to be covered and any covered path element. To create a more detailed and differing rating of generated test cases, the branch distance reflects how far the test object's execution has been from taking the opposite decision at the covered branch node, which is the closest to the structural element to be covered. This approach is suitable for structural testing of program code, like C or Java code.

B. Application to Dynamic Systems

As model-based development is now established in the automotive industry and practitioners have noticed opportunities to test earlier, Windisch [7] as well as Zhan and Clark [6] have transferred the idea of search-based structural testing from code to model level. For SL models, structural coverage criteria similar to the ones known from code testing exist and are commonly accepted in practice. Before addressing the challenges of applying search-based test data generation to SL models, we give a brief introduction to SL. SL is a graphical data-flow language for specifying the behavior of dynamic systems. Syntactically, a SL model consists of functional blocks and lines connecting them, while most of the blocks are equipped with one or more input ports as well as output ports. The semantics of such a model results from the composed functionalities of the involved block types, e.g., sum blocks, relational blocks or delay functions. In addition, event-driven or state-based functionalities can be realized within SL models using SF blocks. A SF block contains an editable Statechart-like automaton.

When applying search-based structural testing to SL models, two fundamental differences compared to its application on code level arise. First, SL models describe time and state dependent processes. Inputs and outputs of SL models, as well as block-connecting lines, are in fact signals. In order to enable reaching all system states, an execution with input sequences (signals) instead of single input values is required. Such complex test data can only be generated with common search algorithms by compressing the data structure, as done by Windisch [13]. His segment-based signal generation approach also considers the necessity for being able to specify the test data signals to be generated (e.g., amplitude bounds and signal characteristic, like wave or impulse form). Second, the aforementioned fitness function approach cannot be fully adopted since SL models are data flow-oriented. There are no execution paths because

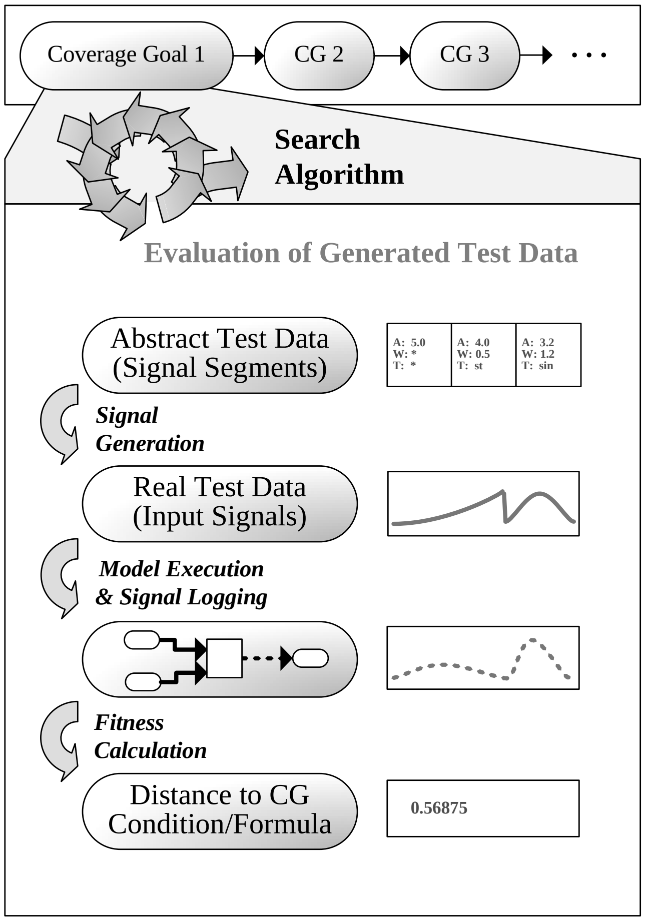

Coverage Goal 1
CG 2
CG 3
. . .
Search Algorithm
Evaluation of Generated Test Data
Abstract Test Data (Signal Segments)
A: 5.0 W: * T: *
A: 4.0 W: 0.5 T: st
A: 3.2 W: 1.2 T: sin
Signal Generation
Real Test Data (Input Signals)
Model Execution & Signal Logging
Fitness Calculation
Distance to CG Condition/Formula
0.56875

Figure 1. Automated search for test sequences, which fulfill coverage goals derived from the model under test.

the execution of a SL model involves the execution of every included block. Hence, a CG-related fitness function addresses only distances to the desired values of one or more model internal signals. For CGs in SF diagrams however, a bipartite fitness approach is possible [8]. Regardless of this, a fitness function has to operate on a sequence (signal) of distance values since distance calculations are done for every time step of the model's execution. Thus, the minimum value of a fitness signal is usually taken as final fitness value.

Figure 1 visualizes the overall work flow of applying search-based test data generation to structural testing of SL models as described.

C. Deficiencies and Potential

Search-based structural testing has been applied successfully to real (proprietary) SL models originating from development projects at Daimler, e.g., a model of a windscreen wiper controller [8]. Compared to purely randomized test data generation of similar complexity, the search-based approach results in significantly higher model coverage. Even in comparison with a commercial tool, the search-based approach performs more effectively.

Despite promising results, the approach lacks efficiency. In general, the overall runtime of the search processes for achieving maximal model coverage increases with the size of the model under test. Similar experiences have been made with code-level search-based structural testing [14]. Since a single automotive SL model is often hundreds of blocks in size, and because a test data generation process of more than a couple of hours is undesirable, improving efficiency of the search-based approach is vital. The following shortcomings of the search-based testing approach for SL, as proposed by Windisch [8], were identified as contributors to efficiency problems. They will be addressed by the work presented in this paper.

1) Even if a model is implementing its desired functionality entirely correctly, there are often model states that are simply unreachable. A search for test data that results in such model states cannot possibly succeed. However, the search technique is not aware of this and carries out a pointless and time-consuming search process.
2) Narrowing the size of the search space, i.e., the space of all possible input data, is crucial for how easy or difficult it is for a search to succeed. Industrial-sized SL models usually have many inputs for which suitable signals need to be found. The size of the overall search space varies with the number of model inputs. Reaching certain model states, however, is often independent of the stimulation of some of the model's inputs.
3) Targeting coverage criteria implies having to reach various CGs, between which, in fact, logical dependancies exist. The coverage of a CG often implies the coverage of other CGs (collateral coverage). In principle, each CG requires a separate search. Those search processes, however, are carried out in an uncontrolled order, regardless of how much collateral coverage they might cause.

There are two further technical problems leading to a lack of efficiency which are only partially addressed by the work presented in this paper. First, the structural test data generation is performed black-box-like, which means that the model is fed with input values on one end while some distances for calculating fitness are measured at some other point in the model. Any structural information between is not considered, thus the search might be blind to complicated dependencies in the model (cf. [15]). Second, when targeting a Boolean state in a model, a suitable fitness function is hard to find since a simple true or false rating inadequately leads a search [8]. Zhan and Clark suggest a technique called tracing and deducing [16], which mitigates this problem in certain cases, but fails in instances where the Boolean problem

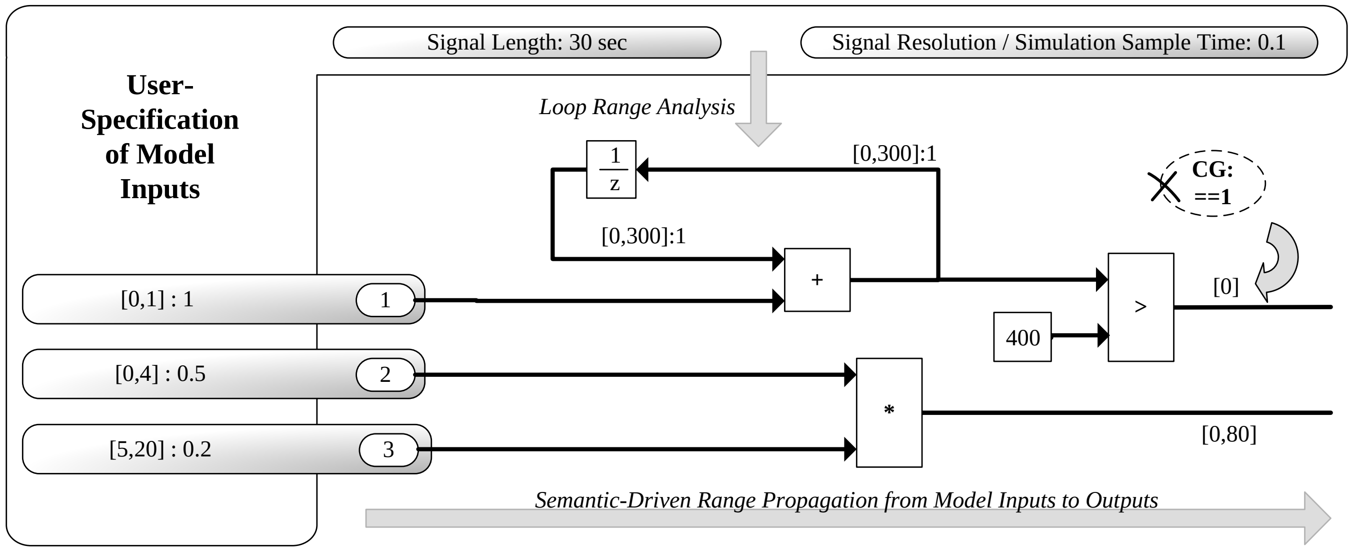

Signal Length: 30 sec
Signal Resolution / Simulation Sample Time: 0.1
User-Specification of Model Inputs
Loop Range Analysis
1/z
[0,300]:1
[0,300]:1
CG: ==1
+
>
[0]
400
[0,1] : 1
1
[0,4] : 0.5
2
*
[5,20] : 0.2
3
[0,80]
Semantic-Driven Range Propagation from Model Inputs to Outputs

Figure 2. Example of how determining the ranges of a model's internal signals based on input specification and block semantics works.

cannot be traced back in the model to a non-Boolean one.

As a whole, we aim to improve the search-based approach for structural testing of SL models so that it performs acceptably and reliably in industrial development environments. To this end, we turn our attention to testing of TL-compliant SL models since the code generator TL is widely used in industrial practice. TL extends SL by offering additional block types, but also makes restrictions on the usage of certain SL constructs like block types. Nevertheless, it is possible to adapt our ideas to pure SL usage.

III. Static Preprocessing

We distinguish between techniques which support search-based structural testing (a) before the CG-related search processes, (b) between the different search processes, (c) during each search process, and (d) after the search processes are done. In the following sections, three techniques belonging to category (a) are presented. Apart from making use of an input specification and choice of coverage criteria provided by the user, all three techniques are fully automatic.

A. Signal Interval Analysis

In structural testing practice, achieving 100% coverage is often not possible. One reason lies in the semantic constructions precluding certain states or signal values. It might also be that a tester specifies the test data to be generated in such a way that it prevents certain CGs from being satisfiable. Also, SL models might be designed variably, e.g., contain a constant block with a variable value. When such variability is bound during execution, e.g., via configuration file, certain model states may be unreachable.

CGs referring to unreachable states worsen the overall runtime and undermine the efficiency of search-based structural test data generation since time-consuming search processes are carried out without any hope of finding desired test data. Therefore, we propose two techniques contributing to automatic identification of unreachable CGs. The first one is an interval analysis, which determines the range within which the values of every internal model signal are. If a signal range is in conflict with the range or value required by a CG, this CG is unsatisfiable. We use interval analysis since other approaches to detect infeasibility, such as constraint solving or theorem proving [17], are currently not scalable enough for the complex equations constituted by industrial-sized SL models. The second technique is an analysis of dependencies between CGs. Since this technique is mainly used for another purpose, it is presented in Section III-C.

The code generator TL, as well as the latest version of SL, are capable of analyzing signal ranges in order to perform code optimizations and improve scaling or data type selection, respectively. While those range analysis features are limited (e.g., determining ranges of signals that are involved in loops is not possible without user interaction) our signal interval analysis (SIA) makes use of an input signal specification in order to overcome such limitations and derive more precise ranges.

As mentioned in Section II-B, a tester who uses the search-based approach for testing SL models, as outlined by Windisch, is asked to specify the test data to be generated first. This involves establishing (a) the range boundaries and step size of each model input, as well as defining (b) a common length (in seconds) and sample rate for all input signals - $sig_{Length}, sig_{Res} \in \mathbb{R}^+$, with sig_{Length} being a multiple of sig_{Res}. SIA starts with information (a) for the model's input signals and propagates the corresponding signal ranges of the form $[x, y]$:q, where q (optional) is the step size, through the whole model. For every model internal signal s_i, we keep a list of consecutive time intervals (time phases) of the form $[t_a, t_b]$, where $t_a, t_b \in \mathbb{N}_0$ and $0 \leq t_a \leq t_b \leq (sig_{Length}/sig_{Res})$. Every time interval goes along with an interval set $I(s_i)_{[t_a,t_b]}$ that contains the actual ranges. We use interval sets instead of a single interval per signal, or per time phase, in order to derive more accurate range information - as suggested by Wang et al. [18].

The propagation technique processes all model blocks in a predetermined order, which is equivalent to the block execution order that SL calculates for running a simulation. Each propagation step is based on the semantics of a block and the ranges of its incoming signals. The result of such a step are ranges (intervals) for the outgoing signals of the block. In this context, we derived interval semantics for each block type of TL-compliant SL models using basic concepts of interval arithmetic [19]. This approach can also be described as a form of abstract interpretation.

Example: The original semantics of a *Sum* block with two incoming signals s_1 and s_2 and the outgoing signal s_3 is $s_{3,t} = s_{1,t} + s_{2,t}$, where t is a time step. Given $I(s_1)_{[t_1,t_2]} = \{[a_{1,1}, b_{1,1}], ..., [a_{1,n}, b_{1,n}]\}$ and $I(s_2)_{[t_3,t_4]} = \{[a_{2,1}, b_{2,1}], ..., [a_{2,m}, b_{2,m}]\}$ with equal or overlapping time intervals, basic interval arithmetic is used to obtain the corresponding interval set for s_3: $I(s_3)_{[max(t_1,t_3),min(t_2,t_4)]} = \{[a_{1,1}+a_{2,1}, b_{1,1}+b_{2,1}]$, ..., $[a_{1,1}+a_{2,m}, b_{1,1}+b_{2,m}]$, ..., $[a_{1,n}+a_{2,1}, b_{1,n}+b_{2,1}]$, ..., $[a_{1,n}+a_{2,m}, b_{1,n}+b_{2,m}]\}$. One could, however, calculate only one resulting interval with the overall minimum and maximum boundary values of the intervals listed above in $I(s_3)$. Since this would lead to a loss of precision, we avoid this approach. In order to still keep interval sets small, we developed an algorithm that merges intervals of the very same set in a suitable way, where possible.

Figure 2 graphically depicts the overall procedure with the aid of a simple example. Note that the model contains a loop, initiated by a delay block with an initial value of 0. The standard propagation procedure would be unable to continue here since ranges are not available for all incoming signals of the sum block. A simple, yet imprecise solution is to set the range of the sum block's outgoing signal to the minimum and maximum values of the signal's data type. A more precise solution, however, is to use the information (b) of the signal specification in order to run a loop analysis. From length and sample rate, the number of loop iterations is derivable. Starting with the initial value of the delay block a static analysis of the loop iterations is performed, resulting in time-related range information. In order to keep the final results clean and minimal, as mentioned before, each signal's ranges as well as the time phases of ranges are combined, if possible, in each iteration of the loop analysis. Note that the ranges in Figure 2 are displayed simplified and summarized, omitting time intervals and the details of their interval sets.

Using range propagation and loop analysis in combination, SIA is capable of determining the ranges of all signals contained in the model under test. In cases of blocks with unknown semantics or unsupported blocks, the minimum and maximum values of the outgoing signal's data type are used. In the end, the results of SIA are used to assess whether each CG's associated formula is unsatisfiable - such as the CG in Figure 2. In addition to unsatisfiable CGs, SIA can also help in identifying Boolean signals or discrete signals with only a few possible different values. As described in Section II-C, CGs related to such signals can be problematic for the search-based approach.

B. Signal Dependency Analysis

By default, the search algorithm generates test data for all of the model's inputs when targeting a CG. However, there are usually CGs whose satisfaction is, in fact, independent of the stimulation of certain model inputs. By not taking this into account, the search space is unnecessarily large, which makes it more difficult for the search to find desired test data. To raise efficiency, we include a signal dependency analysis (SDA) to identify which model inputs each CG actually depends on. McMinn et al. [20] investigated a related approach, however, on code level. SDA is closely related to a slicing approach for SL models developed parallel to our work [21].

At code level, such analysis is usually done by capturing the control dependence in a graph. SL models though, as pointed out previously, are dominated by data dependencies. We therefore analyze the dependency of CGs on input signals by creating a signal dependency graph (a) based on the syntax of the model and (b) refined according to the semantics of blocks. Focusing purely on syntax, the following principle leads to a graph describing which signal b the value of a model internal signal a depends on: Signal a is dominated by a signal b if signal a is the outcome of a model block which has signal b incoming - written $a \rightarrow b$. Some blocks with multiple outgoing signals however, do not use every incoming signal in order to calculate the value of a certain outgoing signal. In such cases, the signal dependency graph is refined by removing over-approximated dependencies. Similar to SIA, SDA processes all model blocks in a predetermined order for collecting dependency relations.

In addition to the basic procedure of SDA as outlined above, some modeling constructs available in SL require special handling. The concept of vector signals, for example, which makes a signal being a container for a number of subordinate signals, requires tracing the dependency between subordinate signals of different containing signals. At this point, the semantics of blocks that process vector signals is relevant. A *Sum* block which has two vector signals as inputs, for example, performs a pair-wise addition of vector elements at the same vector index. Consequently, the resulting signal is also a vector. Each of its subordinate signals, however, is dominated by only two subordinate

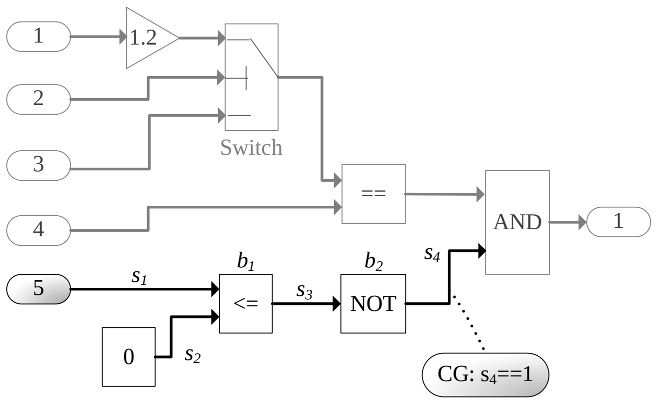


Figure 3. Illustration of how determining which model inputs a CG depends on might narrow the search space.

signals of the block's input signals. SDA's block-specific collecting of dependency relations considers such cases. A further exception that requires special handling by SDA is the concept of conditional subsystems in SL. Such subsystems are only executed if their control signals activate them or keep them active. Every signal inside of such a subsystem thus also depends on the control signal. SDA recognizes these situations when processing the *Inport* blocks within a conditional subsystem and adds dependency relations to the graph accordingly.

In order to finally determine which model inputs a certain CG depends on, the signal or signals which the CG expression refers to are selected in the dependency graph. By traversing the graph up to the input signals, the set of relevant model inputs is built up. Within the subsequent search process for this CG, signals are generated only for the relevant inputs. In addition, operators of the applied search algorithm which merge and modify generated solutions, such as crossover and mutation operators in a genetic algorithm, target only relevant input signals. In this way, depending on the specific application case, the search space might be reduced by several dimensions. For model execution, all other (irrelevant) model inputs receive a random signal that is consistent with the input's specification.

Figure 3 shows an example illustrating the beneficial potential of such an approach in the context of search-based test data generation. SDA's block analysis would build up a dependency graph expressing that $s_3{\rightarrow}s_1$, $s_3{\rightarrow}s_2$ (gathered by analyzing b_1), and $s_4{\rightarrow}s_3$ (gathered by analyzing b_2). These relations alone are sufficient to discover that CG $s_4{==}1$ depends solely on input 5. Hence, varying signals for input 1 to 4 during a search has no effect on satisfying the targeted CG. If the search would focus on exploring the search space of input 5, however, it would likely find the desired input data sooner.

C. Coverage Goal Sequencing

No matter if structural testing is performed in addition to functional (black-box) testing or purely as white-box testing, it is usually a set of CGs that constitutes the test objective. Remember, that for each CG a separate search needs to be run. In Windisch's approach, those search processes are executed in random order. Hence, correlations between CGs are ignored. Given CGs with the expressions $s{<}90$, $s{<}80$ and $s{<}70$, for example, it is most likely more efficient to aim for reaching the goal $s{<}70$ first because it satisfies all other CGs at the same time. As this example indicates, the execution order of the CG-related search processes affects the efficiency of the whole structural test.

Other researchers in the search-based testing community have noticed this shortcoming as well. Fraser and Arcuri [22] advise focusing on the generation of whole test suites rather than targeting single CGs. They recommend optimizing multiple test suites instead of multiple test data and also suggest rewarding smaller test suites with a better fitness, in case two or more test suites achieve the same coverage. Harman et al. [23], in contrast, suggest a multi-objective search in which each CG is still targeted individually but the number of collateral (accidentally covered) goals is included as a secondary objective. Though facing a similar problem, our approach differs. We keep the focus on CGs themselves since, considering the complexity of the optimization problems constituted by industrial SL models, they are often difficult to reach and we do not want to impede the search by burdening it with additional goals or mixed fitness values. Instead, we propose a coverage goal sequencing (CGSeq) approach that creates a reasonable order in which the various CGs are pursued. Li et al. worked out a related approach [24], however, it is outside of the search-based and SL context. Ultimately, by maximizing collateral coverage, our approach attempts to minimize the number of CGs that need to be pursued. Not only is this expected to improve overall efficiency, but the resulting test suite should also be smaller.

The procedure of CGSeq is summarized in Figure 4. First of all, the model under test is analyzed and CGs are derived for all SL/SF-relevant coverage criteria (see overview by Windisch [8]). In preparation to analyzing dependencies between CGs, we apply several harmonization and simplification steps to the CG expressions. Note that results of SIA (Section III-A) are used for this task as well, e.g., an expression $s{\geq}1$ would be transformed to $s{=}1$ if s is a Boolean signal.

Next, possible dependencies between CG expressions are analyzed, resulting in a dependency graph. In this graph, we treat the nodes as CG sets, in which equivalent CGs are grouped - noted as $CGN = \{CG_a, ..., CG_n\}$. Note that we omit the set braces in some of the following notations if the set contains only one element. In order to limit graph complexity, the dependency analysis considers only

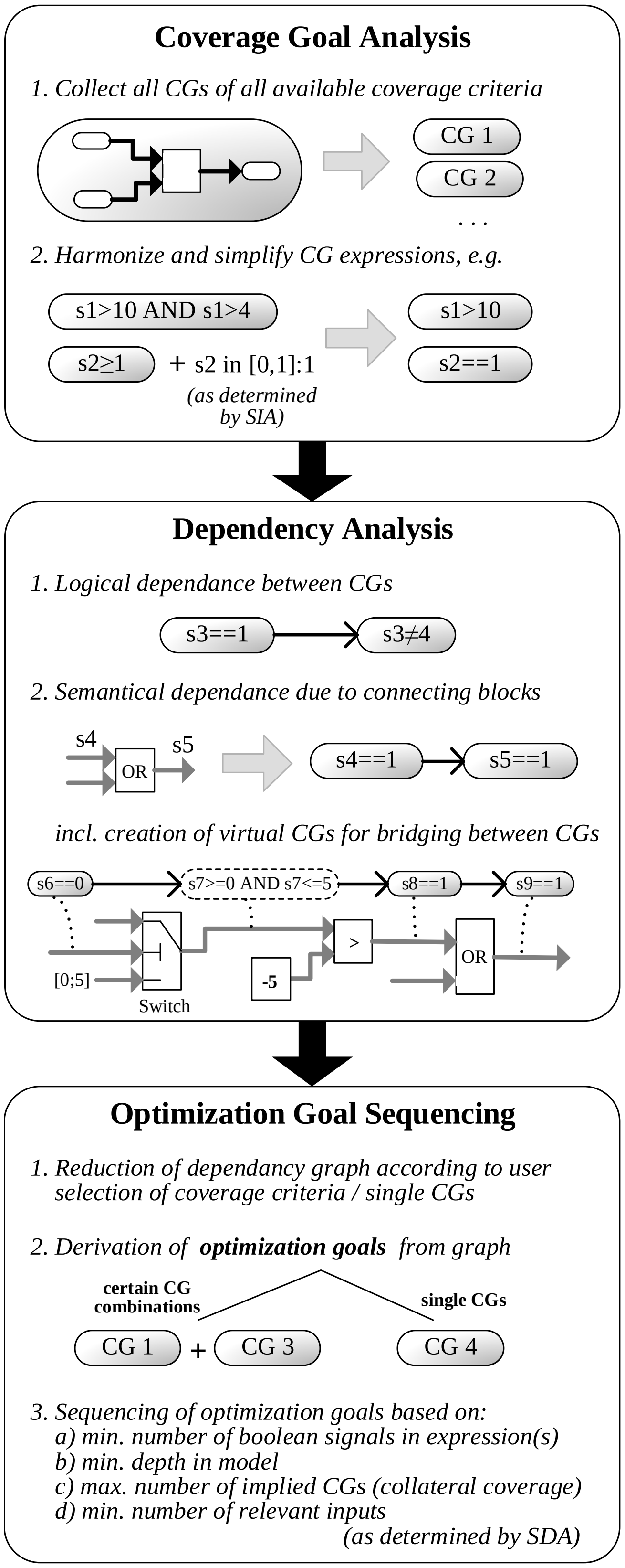

Coverage Goal Analysis
1. Collect all CGs of all available coverage criteria
CG 1
CG 2
...
2. Harmonize and simplify CG expressions, e.g.
s1>10 AND s1>4
s1>10
s2≥1
+ s2 in [0,1]:1
(as determined by SIA)
s2==1
Dependency Analysis
1. Logical dependance between CGs
s3==1
s3≠4
2. Semantical dependance due to connecting blocks
s4
s5
OR
s4==1
s5==1
incl. creation of virtual CGs for bridging between CGs
s6==0
s7>=0 AND s7<=5
s8==1
s9==1
[0;5]
Switch
-5
>
OR
Optimization Goal Sequencing
1. Reduction of dependancy graph according to user selection of coverage criteria / single CGs
2. Derivation of optimization goals from graph
certain CG combinations
single CGs
CG 1
+
CG 3
CG 4
3. Sequencing of optimization goals based on:
a) min. number of boolean signals in expression(s)
b) min. depth in model
c) max. number of implied CGs (collateral coverage)
d) min. number of relevant inputs
(as determined by SDA)

Figure 4. Main process steps to create an efficient order for a set of coverage goals which are processed separately by a search.

relations that are useful for assessing CG satisfiability and for the final goal of sequencing the CGs. The following relations are captured: implication ($CGN_1 \rightarrow CGN_2$), equivalence ($CGN_1 \cup CGN_2$), NAND ($CGN_1 \uparrow CGN_2$), and XOR ($CGN_1 \oplus CGN_2$). For a more compact notation of implications we also use, in certain cases, conjunctions ($(CGN_1 \wedge ... \wedge CGN_n) \rightarrow (CGN_m \wedge ... \wedge CGN_z)$) and disjunctions ($(CGN_1 \vee ... \vee CGN_n) \rightarrow (CGN_m \wedge ... \wedge CGN_z)$). Dependencies between CG expressions are analyzed both from a logical and a semantical point of view.

Logical dependency addresses relations between CGs due to the operators involved in their formulas, as well as due to contained constants and the ranges of related signals, if SIA has been carried out prior to CGSeq. For example, the relation $(s_1{>}0)_{CG} \rightarrow (s_1{\geq}0)_{CG}$ is recognized solely based on the involved operators. The relation $(s_1{==}5)_{CG} \rightarrow (s_1{\geq}0)_{CG}$, however, requires taking the involved constants into account. Going one step further, the relation $(s_1{==}3)_{CG} \rightarrow (s_1{>}s_2)_{CG}$ would be detected if $max(I(s_2)){<}3$, i.e., the maximum upper boundary of any interval in s_2's range set is less than the constant 3. An extensive distinction of such cases has been worked out.

Semantical dependency means that for each two CGs relating to incoming or outgoing signals of the same model block, a block-specific analysis checks if a relation between the CGs exists. Just as within SIA and SDA, all model blocks are processed in a predetermined order for this analysis. Given an *OR* block with the incoming signals s_1 and s_2, and the outgoing signal s_3, the analysis for this block would detect, for example, the relations $(s_1{\neq}0)_{CG} \rightarrow (s_3{==}1)_{CG}$, $(s_2{\neq}0)_{CG} \rightarrow (s_3{==}1)_{CG}$, and $((s_1{==}0)_{CG} \wedge (s_2{==}0)_{CG}) \rightarrow (s_3{==}0)_{CG}$. In certain cases the block-specific analysis adds virtual CGs as a bridge to other CGs in order to detect further dependencies. For this purpose, as illustrated in Figure 4, signal range information from SIA is also used in case of certain block types.

As a side effect, based on the captured dependencies, further CGs might be detected as unsatisfiable in the course of CGSeq (cf. Section III-A). For example, if the graph contains the relation $CG_1 \leftrightarrow CG_2$ and SIA has discovered that CG_2 is unsatisfiable, then CG_1 must also be unsatisfiable. CGSeq performs an extended satisfiability check by propagating unsatisfiability conclusions through the graph.

Now, back to the goal of sequencing the CGs. In the next step, the user's selection of coverage criteria or single CGs is considered by minimizing the graph accordingly. Amongst others, non-selected CGs implying selected CGs are kept. Afterwards, the final optimization goals are derived from the graph in a two-fold way:

1) For each selected CG, called $CG_{Original}$, an optimization goal consisting of a single CG, called CG_{Target}, is derived. Starting in the dependency graph at the node of $CG_{Original}$, the implication relations are analyzed backwards to determine CG_{Target}, which

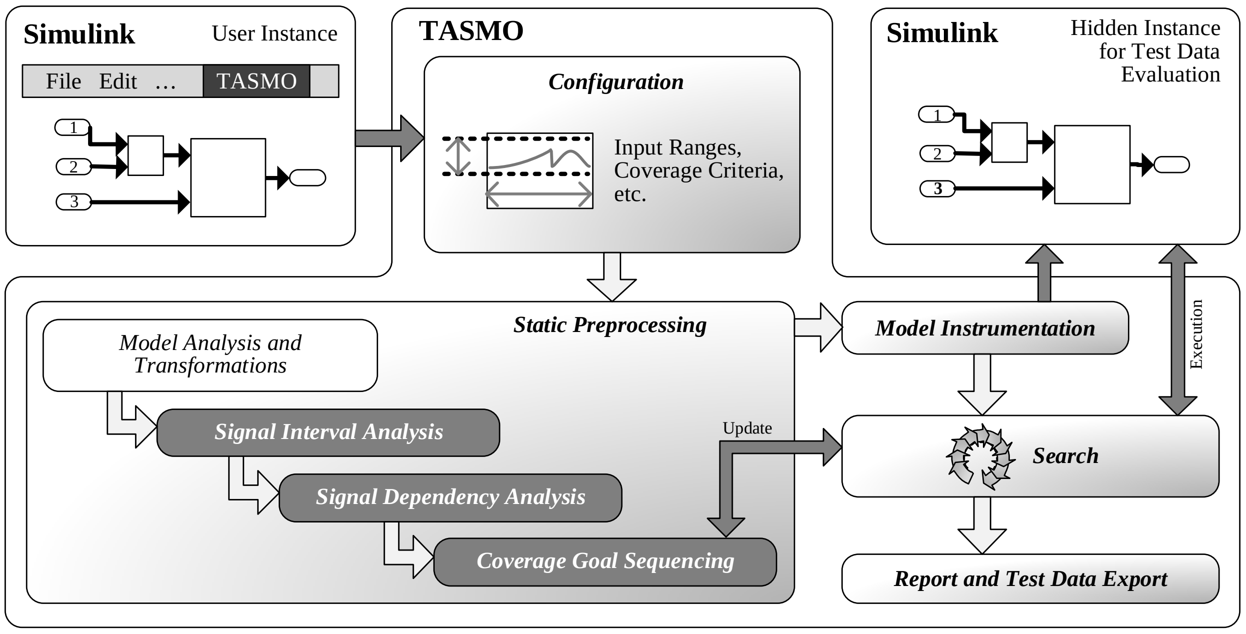


Figure 5. Overall workflow of the tool prototype for search-based test data generation for Simulink, including the static preprocessing workflow.

ultimately is representing $CG_{Original}$. If multiple CGs are suitable candidates for CG_{Target}, criteria such as a lower model depth (minimum path length from any model input to any signal involved in the CG) or a lower rate of Boolean signals involved in the CG expression are taken into account.

2) Certain combinations of CGs are derived to form optimization goals as well. Remember that the dependency graph can contain conjunctions. Each conjunction serves as a start point for building up a conjunction tree which contains implication relations from the graph that are leading (directly or indirectly) to the conjunction. Out of every branch or sub-branch within this tree one suitable CG is selected. These CGs constitute one or multiple new optimization goals, depending on the depth of the tree. In this way, a few suitable optimization goals with potentially high collateral coverage are added.

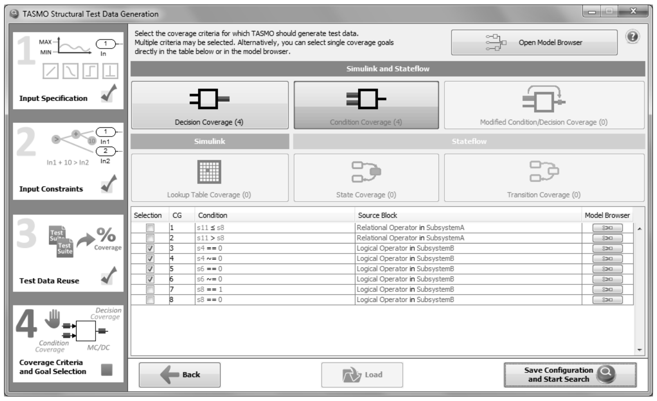
Finally, the optimization goals are sequenced according to several metrics, primarily by the number of (so far unsatisfied) implied CGs, but also by their depth in the model and the amount of Boolean signals involved in the expressions - since such goals should be avoided given the fitness function construction problem (see Section II-C). Note that the pursuing order of optimization goals is updated after each search process ends, since an optimization goal's number of unsatisfied implied CGs might have changed.

IV. Implementation

The presented preprocessing techniques have been implemented in the course of developing our prototypical tool *TASMO* (Testing via Automated Search for Models) [25].

TASMO is mainly written in Java and closely integrated with Matlab. As shown in Figure 5, the user can trigger the automated test data generation process directly from within Matlab when having a SL model opened. *TASMO* then extracts model related information using Matlab's API and programming language *m*. After the Java-based component of *TASMO* has been triggered, it builds up an internal representation of the model under test and derives all CGs for every supported coverage criteria from this representation in advance. It then applies transformation and reduction steps to the internal model representation in order to focus on the relevant parts for structural test data generation. In particular, all blocks and signals from the model's *Outports* to the rightmost CGs are removed since they are irrelevant for the following analysis and preprocessing steps. Model constructs like virtual subsystems or bus systems, which are both semantically irrelevant, are flattened.

The user is then asked to select one or more coverage criteria, like decision coverage or condition coverage, or certain CGs directly. Regarding this step, an insight into the user interface of the tool is given in Figure 6. As indicated earlier, the user is also asked to specify the test data to be generated, e.g., the range of each model input. After all required presettings have been made, the three presented preprocessing techniques are run in the following order: First, SIA determines the ranges of all model internal signals in order to assess the satisfiability of each CG. During SIA, *TASMO* might also transform the internal model representation. Both general and block-specific transformation



Select the coverage criteria for which TASMO should generate test data.
Multiple criteria may be selected. Alternatively, you can select single coverage goals directly in the table below or in the model browser.

Selection	CG	Condition	Source Block	Model Browser
☐	1	s11 ≤ s8	Relational Operator in SubsystemA	
☐	2	s11 > s8	Relational Operator in SubsystemA	
✓	3	s4 == 0	Logical Operator in SubsystemB	
✓	4	s4 ~= 0	Logical Operator in SubsystemB	
✓	5	s6 == 0	Logical Operator in SubsystemB	
✓	6	s6 ~= 0	Logical Operator in SubsystemB	
☐	7	s8 == 1	Logical Operator in SubsystemB	
☐	8	s8 == 0	Logical Operator in SubsystemB	

Figure 6. Automated testing of Simulink models with *TASMO*: Selection of coverage criteria and coverage goals for the model under test.

criteria are checked. For example, if a signal's ranges all contain only a single constant value and this signal does not originate from a *Constant* block, the source block of this signal is replaced with a *Constant* block. At the same time, a backwards directed removal process is initiated starting with the block that has been replaced. All signals and blocks preceding this block (directly or indirectly) are removed if no CGs are referenced to it. In this way, the internal model is kept as compact as possible for subsequent analysis steps without altering its semantics. Another example: If the incoming signal of an *Abs* block (absolute value) has solely non-negative ranges, the block can be removed and its former incoming signal can be directly connected to its former outgoing signal.

After SIA, *TASMO* runs SDA in order to determine for each CG which model inputs it depends on. Finally, CGSeq is carried out to build up and sequence a list of optimization goals. Before the search algorithm processes optimization goal after optimization goal, a hidden Matlab instance for simulating the model with generated test data is started. A copy of the real model under test is loaded, initialized, and instrumented. Afterwards, an automated search for each optimization goal is performed, which generates test data solely compliant with the signal specification for all inputs that are relevant for the CGs of the current optimization goal. If an optimization goal, or rather its contained CGs, are covered by accident during a search for another optimization goal (collateral coverage), the goal is considered done. When finished, *TASMO* generates a report and provides the generated test data in a reusable format.

V. Case Study

We investigated the effect of the three preprocessing techniques SIA, SDA and CGSeq on structural test data generation for industrial SL/TL models. Two SL/TL models served as case examples, model A and B.

Model A originates from the development of an electric vehicle's propulsion strategy, contains 730 blocks, and has 12 inputs. A total of 600 CGs were derived for the chosen coverage criteria *decision coverage* and *condition coverage*. Model B implements the functionality of a rear window defroster. It contains 1861 blocks, has 16 inputs, and 1113 CGs were derived in total.

Table I
Case Study Configurations

Configuration	SIA	SDA	CGSeq
C1	-	-	-
C2	x	-	-
C3	x	x	-
C4	x	-	x
C5	x	x	x

In order to analyze the effect of the presented techniques, five different configurations were compared. Table I outlines the characteristics of each configuration. The configurations differ in the use or non-use of the static preprocessing techniques within the test data generation process. C2 is compared with C1 to evaluate the effect of SIA, C3 is compared with C2 to evaluate the effect of SDA, and C4 is compared with C2 to evaluate the effect of CGSeq. C5 brings all three static preprocessing techniques together to evaluate the entire static preprocessing as a whole. In particular, only C5 allows CGSeq to use the results of both SIA and SDA in order to analyze dependencies between the coverage goals and to derive an execution order for the search goals. Note that SIA is also activated in C3 and C4, since processing unsatisfiable CGs would otherwise extend the runtime of the case study unnecessarily.

For the search, we applied a special genetic algorithm for generating signals, as presented by Windisch and Al Moubayed [13] (see Section II-B). The algorithm settings were chosen as listed in detail in their paper, except for the following differences. In each search iteration, 20 individuals (test data) were generated. A search was stopped when the targeted optimization goal was reached, when 40 iterations had been carried out, or if the search stagnated. Search stagnation was indicated by 8 successive iterations in which no better individual was found. Since the search algorithm is subjected to a certain randomness, we carried out a total of 30 test data generation runs for each configuration (C1-C5). The input signals to be generated for model A were specified to have a length of 30 seconds and a sample rate of 0.1 seconds. For model B, the signal length was set to 20 seconds and the sample rate to 0.25 seconds. While the signal lengths were chosen with respect to the dynamic functional behavior of the models, the sample rates are specified in each model's properties. Range boundaries for each model's inputs (see Section II-B and III-A) were taken directly from development documents.

The study was run on a PC with an Intel Core 2 Duo processor (P8600, 2.4 GHz), 4 GB RAM, and Windows 7

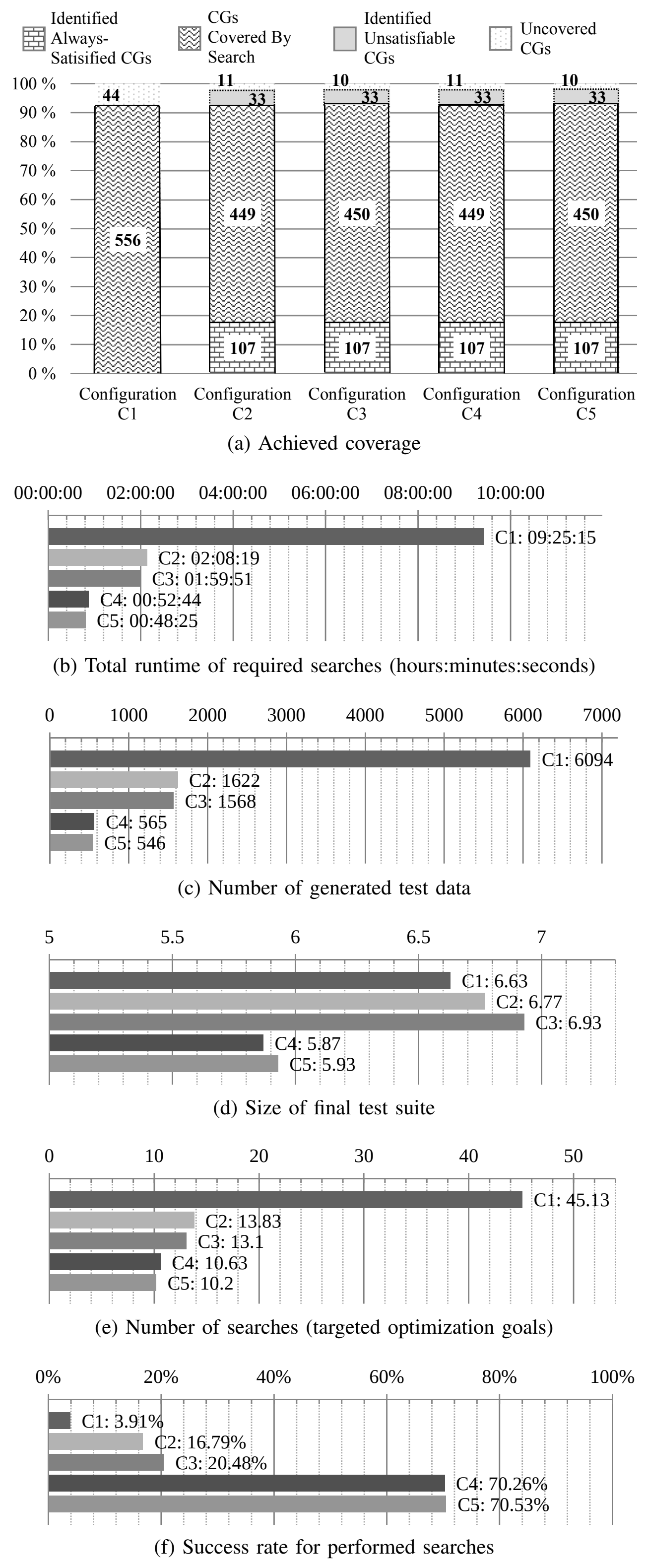


Figure 7. Measured variables of test data generation for model A, averaged over 30 test data generation runs.

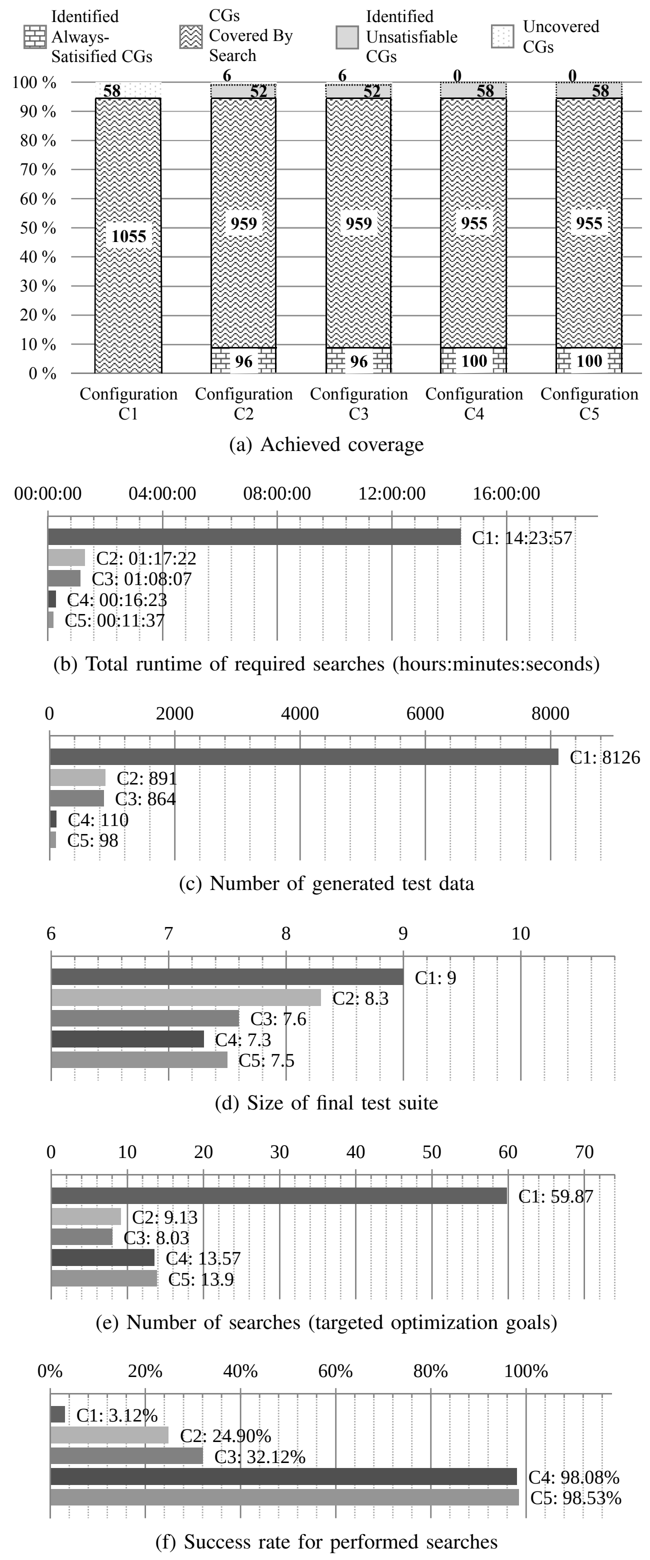


Figure 8. Measured variables of test data generation for model B, averaged over 30 test data generation runs.

64-bit. Our Java-based tool *TASMO* was run using version 7 of the Java Runtime Environment in connection with Matlab/Simulink 2009b and TargetLink 3.1.

In the following sections, the results of the case study are presented. First, we analyze the results with regard to each preprocessing technique separately. Then, we assess the use of our preprocessing techniques as a whole.

A. Analysis of SIA

As visible in Figure 7a for C2, SIA identified 33 unsatisfiable CGs for model A, of which 8 were identified by SIA's loop analysis. In addition, 69 CGs from a total of 107 CGs, which are always satisfied independent of the chosen input data, were identified by SIA. All those CGs were excluded from subsequent search processes. As for model B, 58 CGs are unsatisfiable, of which 52 were identified by SIA (6 by its loop analysis), as can be seen for C2 in Figure 8a. For model B, SIA also identified 96 CGs from a total of 100 CGs that are always satisfied. Note that the other always-satisfied or unsatisfied CGs were identified by CGSeq.

In order to evaluate the effect of preprocessing the test data search with SIA, C1 and C2 are analyzed. The model coverage achieved by a configuration is used to measure for effectiveness of a configuration. C1 was able to generate test data for model A with a coverage of almost 93% (95% for model B). While introducing SIA in C2 did not lead to higher coverage, as can be seen in Figures 7a and 8a, it certainly improved the efficiency of the automated test data generation process. As the statistics for model A in Figure 7b show, the runtime of all performed searches together was reduced by 77% (from 9:25 hours for C1 to 2:08 hours for C2). For model B, the runtime was even reduced by 91% (from 14:24 hours for C1 to 1:17 hours for C2, see Figure 8b). The runtime of SIA itself, which is not included in the values displayed in Figures 7b and 8b, was about 5 minutes for model A and 15 seconds for model B. In large part, SIA's runtime was caused by its loop analysis feature.

In C2, test data was only generated for CGs that were not identified as unsatisfiable. While the test data generation process of C1 for model A was targeting about 45 CGs directly by search (about 60 for model B), only 14 (9 for model B) were targeted by C2 (see Figures 7e and 8e). The number of searches that were carried out additionally by C1 is approximately as high as the number of unsatisfiable CGs that SIA was able to identify. As a consequence of C2 targeting less CGs, less test data was generated overall, as displayed in Figures 7c and 8c. Since C1 performed searches for unsatisfiable CGs without any hope for success, the portion of successfully tackled CGs is naturally higher for C2 in comparison to C1 (see Figures 7f and 8f).

B. Analysis of SDA

Using SDA, the search processes of C3 and C5 were ignoring irrelevant inputs during input data optimization. According to the preprocessing analysis carried out by SDA, on average, 4.18 of model A's 12 inputs turned out to be relevant for reaching one of the CGs (6.58 of 16 inputs for model B). Considering only the CGs that were targeted in C3, 6.48 of the 12 inputs of model A, on average, are relevant (12.06 of 16 for model B). For C5, the average value is 4.71 (model A) and 7.85 (model B), respectively. These values reflect the effective search space reduction due to the use of SDA.

Comparing the coverage results of C3 with those of C2, as well as C5 with C4, the introduction of SDA to the automated test data generation process for model A led to slightly higher coverage (see Figure 7a). One coverage goal that was only covered seldomly by the other configurations was always covered with SDA being activated. This observation is backed up by the increased rate of successfully finished searches for C3, as visible in Figure 7f. In case of model B, C1 managed to cover as many CGs as C2.

The use of SDA reduced the overall runtime of the searches by 7% for model A (from 2:08 hours for C2 to 1:59 hours for C3) and 12% for model B (from 1:17 hours for C2 to 1:08 hours for C3), as visible in Figures 7b and 8b. For both models, the runtime of SDA itself was only about one second. Due to focusing solely on relevant model inputs, the searches for a few targeted CGs finished slightly quicker. Accordingly, less test data was generated, as can be seen when comparing C3 with C2 in Figures 7c and 8c.

C. Analysis of CGSeq

Note that during CGSeq, another 38 always-satisfied CGs were identified for model A in addition to 69 such CGs that were already identified by SIA (107 in total as shown in Figure 7a). Likewise, 4 additional always-satisfied CGs were identified by CGSeq for model B, resulting overall in 100 such CGs (see Figure 8a). All these CGs were considered as covered and were excluded from the test data generation process since any test data in the generated test suite would satisfy them. For model B, CGSeq even found another 6 unsatisfiable CGs, in addition to 52 unsatisfiable CGs identified by SIA (see Figure 8a).

Using CGSeq prior to the test data search (as done in C4) did not increase the total model coverage (compared to C2), as apparent in Figures 7a and 8a. However, the runtime of the search was reduced by 59% for model A (from 2:08 hours for C2 to 52 minutes for C4) and 79% for model B (from 1:17 hours for C2 to 16 minutes for C4) due to introducing CGSeq in the test data generation process, as visible in Figures 7b and 8b. Similar to SIA and SDA, CGSeq itself did turn out to be reasonable in terms of computation time. Running CGSeq took only about 16 seconds for model A and 20 seconds for model B on average.

Due to targeting more promising (e.g., non-boolean) goals first, the portion of successfully tackled searches was higher (see Figures 7f and 8f) and less test data had to be generated (see Figures 7c and 8c). For model A, less searches were performed (compare C4 and C2 in Figure 7e) and the automated test data generation process resulted in smaller test suites (see Figure 7d) since CGSeq prioritizes goals with potentially high collateral coverage. While the size of the resulting test suite was also reduced when activating CGSeq for model B (compare C4 and C2 in Figure 8d), more searches were performed (see Figure 8e). This might be surprising at first sight. However, it demonstrates how the test data generation process becomes more target-oriented when using CGSeq. In C2, a lot of CGs that are difficult to approach for the search algorithm were targeted. Plenty of test data was generated in the course of this and as a side effect, many CGs were covered rather coincidentally.

D. Combined Analysis

In summary, all three preprocessing techniques have demonstrated their usefulness to automated test data generation for SL/TL models. The case studies have shown that SIA is able to raise the test data generation's efficiency significantly, due to exclusion of unsatisfiable CGs. SDA pointed out its capability to raise both effectiveness and efficiency by turning the focus of the search for input data on relevant model inputs. CGSeq improved the efficiency considerably by advising the test data search to preferably target CGs for which a suitable fitness function is derivable, and which entail high collateral coverage. In addition, the runtime of all three preprocessing techniques was vanishingly low compared to the runtime of a subsequent test data search. Thus, we conclude that their use comes without any significant negative side effects.

Running all three prepocessing techniques in combination, as done in C5, indicates that the advantages of each technique also complement each other suitably. While in C3, only SIA and SDA, and in C4, only SIA and CGSeq were combined, the combination of all of them led to a further reduction of the search runtime of 8% for model A and 29% for model B (see Figures 7b and 8b). Even though the improvement in terms of runtime, as well as number of generated test data, performed searches, and ratio of successful searches, is only small (see Figure 7), C5 came out on top of all configurations for both effectiveness and efficiency of the automated test data generation, in case of model A as well as in the case of model B.

The case study also demonstrates the practicability and feasibility of search-based test data generation for SL/TL models, if extended by static preprocessing such as the presented ones. In particular, even if no automated test oracle is available to evaluate the generated test suite's conformance to the specification, and the test suite thus needs to be analyzed manually under a functional aspect, the case study has shown that the applied test data generation approach can lead to relatively small test suites, making a manual analysis feasible. Note that applying test suite minimization techniques might further reduce the size of the obtained test suites.

Further improvements and extensions of the applied test data generation technique might increase its performance to industrial SL/TL models even more. We investigated, for instance, why certain CGs of model A were not covered at all during the case study. It turned out that 8 CGs (of 10 uncovered ones, see Figure 7a) are unsatisfiable and neither SIA nor CGSeq were able to identify this circumstance; mainly due to block types in the chosen model for which the preprocessing techniques did not offer a specific handling. The two satisfiable, yet uncovered CGs left, were found to be difficult to solve by the search-based algorithm. Further guidance, for instance, by advanced fitness functions, is required to facilitate the search to reach such CGs. Nevertheless, the work presented in this paper turned out to be a big step forward towards industrial applicability of search-based automation of structural testing for SL/TL models.

VI. Related Work

Besides search-based testing, as introduced in Section II, a few other approaches have been applied or proposed for automating structural test data generation for SL models. In general, the approaches can be classified as dynamic, static or hybrid techniques. Dynamic techniques, such as search-based test data generation, execute the test object during test data generation. Static techniques however, analyze the test object without executing it in order to generate test data. Hybrid techniques usually contain characteristics of both dynamic and static techniques. Our approach mainly utilizes a dynamic technique, but could also be classfied as a hybrid approach since it is combined with static techniques.

In the field of structural test data generation for SL, the following dynamic techniques have been applied: search-based testing and random testing. Case studies of Windisch have shown that search-based testing outperforms random testing for structural testing of SL models [8]. As for static techniques, the use of symbolic execution/constraint solving and model checking has been reported in the SL context. Gadkari et al. developed an approach called *AutoMOTGen* [26], which involves the transformation of SL/SF models into a representation of the high-level language SAL and the use of a model checker for generating test data [27]. While the authors encountered promising results, they were also faced with technical limitations of their approach, particularly scalability issues. Păsăreanu et al. use symbolic execution and constraint solving in order to generate test data for SL models [28]. They also transform the model first - in their case, into Java code. The tool *Symbolic PathFinder* is then used to generate test data. However, their reports indicate that this approach suffers from scalability issues. Satpathy et al. developed *REDIRECT*, which applies concolic testing to the test data generation problem for SL [29]. Similar to the work of Gadkari et al., they transform

SL/SF models to SAL first. Concolic testing extends test data generation via symbolic execution and constraint solving by random testing. *REDIRECT* could also be classified as a hybrid approach. The authors document satisfying results, however, the SL models used in their case studies focus on SF diagrams and are much smaller than the ones used in this work. Peranandam et al. went along a similar path and combined random testing, constraint solving, model checking, and heuristics in their test data generation tool *SmartTestGen* for SL/SF models [30]. *SmartTestGen* uses a classification algorithm that statically estimates which of the listed techniques is likely to cover the coverage goal in question and therefore applies it for test data generation. The authors conclude that a case-specific use of different test data generation techniques is advantageous to mastering the complexity of industrial-sized SL models and the heterogeneity of their test data generation problems.

Commercial tools for structural test data generation for SL models are available as well. *Reactis* [31] from Reactive Systems, *T-VEC* [32] from T-VEC Technologies, and *Simulink Design Verifier* [33] from The MathWorks are common tools for this job. Only little is published about how these tools generate test data. Windisch [8] asserted that all these tools use randomized test data generation in some way. While *Reactis* combines this basic technique with guided simulations, *T-VEC* with symbolic execution and constraint solving, and *Simulink Design Verifier* with static analysis, the details of their implemented techniques remain unknown.

While dynamic approaches might face efficiency issues at times, as described for search-based testing in Section II-C, or have difficulties covering certain structural goals, as in the case of randomized test data generation, purely static approaches seem to lack scalability. As demonstrated in this paper, we believe in the potential of hybridization, i.e., the extension of dynamic test data generation techniques with static techniques. Since search-based test data generation is basically advanced randomized test data generation, which is used as a basic technique in various approaches and tools, we continue following the search-based testing path. Note that our prototypical tool *TASMO* is also applying different test data generation strategies, such as constraint solving, when a CG is suitable. This option was deactivated during the case studies presented in this paper though. More generally, the presented static preprocessing techniques are even usable independent of the choice of test data generation technique. Identifying unsatisfiable CGs, focusing on relevant model inputs, and generating an order of satisfiable CGs for efficient processing, is also advantageous when the test data is generated by randomization, constraint solving, or hybrid techniques.

VII. Conclusion and Future Work

This paper introduces an approach to improving the performance of search-based testing when applied to structural testing of SL models. Three static techniques extend the standard search-based approach by analyzing the model under test before the search processes for each CG are run. Unsatisfiable CGs are partially identified and excluded from the search. The search space is reduced in such a way that the search focuses solely on relevant model inputs. The separate search processes for each CG are sequenced in order to maximize collateral coverage, minimize test suite size, and shorten the overall search runtime.

A tool prototype that demonstrates the applicability of the extended search-based approach in industry has been developed. A case study with two industrial SL/TL models from the automotive domain has been performed, demonstrating how the presented preprocessing techniques improve the search-based test data generation approach. In particular, efficiency was raised distinctly. The use of all three techniques in combination led to a reduction of over 90% in the runtime otherwise required for search-based test data generation without the use of any of these techniques.

Despite satisfactory results, further adaption of the tool to industrial requirements is required, e.g., improved support of SL/TL blocks and SF diagrams by the presented static preprocessing techniques. Our next main step is to work on an expanded hybridization of the test data generation process. Besides integrating constraint solving techniques, we plan on using different types of search algorithms in combination, while factoring more information collected during model execution into the (hybridized) search algorithm. Finally, a broader comparison of the approach and tool, in particular with established commercial tools, is required.

References

[1] B. Wilmes, “Automated structural testing of Simulink / TargetLink models via search-based testing assisted by prior-search static analysis,” in *VALID 2012, The Fourth International Conference on Advances in System Testing and Validation Lifecycle*, 2012, pp. 51–56.

[2] The Mathworks, “Matlab Simulink,” Last access: 2013-08-16. [Online]. Available: http://www.mathworks.com

[3] dSpace, “Targetlink,” Last access: 2013-08-16. [Online]. Available: http://www.dspace.com

[4] P. McMinn, “Search-based software testing: Past, present and future,” in *IEEE 4th International Conference on Software Testing, Verification and Validation Workshops*, ser. ICSTW ’11, 2011, pp. 153–163.

[5] B. Wilmes, A. Windisch, and F. Lindlar, “Suchbasierter Test für den industriellen Einsatz,” in *4. Symposium Testen im System- und Software Life-Cycle*, 2011.

[6] Y. Zhan and J. A. Clark, “A search-based framework for automatic testing of MATLAB/Simulink models,” *Journal of Systems and Software*, vol. 81, no. 2, pp. 262–285, Feb. 2008.

[7] A. Windisch, "Search-based testing of complex Simulink models containing Stateflow diagrams," in *31st International Conference on Software Engineering*, 2009, pp. 395–398.

[8] A. Windisch, "Suchbasierter Strukturtest für Simulink Modelle," Ph.D. dissertation, Berlin Institute of Technology, 2011.

[9] W. Miller and D. L. Spooner, "Automatic generation of floating-point test data," *IEEE Transactions on Software Engineering*, vol. 2, no. 3, pp. 223–226, May 1976.

[10] B. Korel, "Automated software test data generation," *IEEE Transactions on Software Engineering*, vol. 16, no. 8, pp. 870–879, Aug. 1990.

[11] M. Harman and P. McMinn, "A theoretical and empirical study of search-based testing: Local, global, and hybrid search," *IEEE Transactions on Software Engineering*, vol. 36, pp. 226–247, 2010.

[12] J. Wegener, A. Baresel, and H. Sthamer, "Evolutionary test environment for automatic structural testing," *Information and Software Technology*, vol. 43, no. 14, pp. 841–854, 2001.

[13] A. Windisch and N. Al Moubayed, "Signal generation for search-based testing of continuous systems," in *International Conference on Software Testing, Verification and Validation Workshops*, ser. ICSTW '09, 2009, pp. 121–130.

[14] T. E. Vos, A. I. Baars, F. F. Lindlar, P. M. Kruse, A. Windisch, and J. Wegener, "Industrial scaled automated structural testing with the evolutionary testing tool," in *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation*, ser. ICST '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 175–184.

[15] P. McMinn, M. Harman, D. Binkley, and P. Tonella, "The species per path approach to search based test data generation," in *Proceedings of the 2006 International Symposium on Software Testing and Analysis (ISSTA)*, ser. ISSTA '06. New York, NY, USA: ACM, 2006, pp. 13–24.

[16] Y. Zhan and J. A. Clark, "The state problem for test generation in Simulink," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2006, pp. 1941–1948.

[17] A. Goldberg, T. C. Wang, and D. Zimmerman, "Applications of feasible path analysis to program testing," in *Proceedings of the International Symposium on Software Testing and Analysis*, ser. ISSTA '94. New York, NY, USA: ACM, 1994, pp. 80–94.

[18] Y. Wang, Y. Gong, J. Chen, Q. Xiao, and Z. Yang, "An application of interval analysis in software static analysis," in *Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, ser. EUC '08, vol. 2. Washington, DC, USA: IEEE Computer Society, 2008, pp. 367–372.

[19] R. E. Moore, *Interval Analysis*. Prentice-Hall, 1966.

[20] P. McMinn, M. Harman, K. Lakhotia, Y. Hassoun, and J. Wegener, "Input domain reduction through irrelevant variable removal and its effect on local, global, and hybrid search-based structural test data generation," *IEEE Transactions on Software Engineering*, vol. 38, pp. 453–477, 2012.

[21] R. Reicherdt and S. Glesner, "Slicing Matlab Simulink models," in *34th International Conference on Software Engineering*, 2012, pp. 551–561.

[22] G. Fraser and A. Arcuri, "Evolutionary generation of whole test suites," in *11th International Conference on Quality Software*, 2011, pp. 31–40.

[23] M. Harman, S. G. Kim, K. Lakhotia, P. McMinn, and S. Yoo, "Optimizing for the number of tests generated in search based test data generation with an application to the oracle cost problem," in *3rd International Conference on Software Testing, Verification, and Validation Workshops*, ser. ICSTW '10, 2010, pp. 182–191.

[24] J. J. Li, D. Weiss, and H. Yee, "Code-coverage guided prioritized test generation," *Information and Software Technology*, vol. 48, no. 12, pp. 1187–1198, 2006.

[25] B. Wilmes, "Toward a tool for search-based testing of Simulink/TargetLink models," in *4th Symposium on Search Based Software Engineering (Fast Abstracts)*. Fondazione Bruno Kessler, 2012, pp. 49–54.

[26] A. A. Gadkari, A. Yeolekar, J. Suresh, S. Ramesh, S. Mohalik, and K. C. Shashidhar, "AutoMOTGen: Automatic model oriented test generator for embedded control systems," in *Proceedings of the 20th International Conference on Computer Aided Verification*, 2008, pp. 204–208.

[27] A. A. Gadkari, S. Mohalik, K. Shashidhar, A. Yeolekar, J. Suresh, and S. Ramesh, "Automatic generation of test-cases using model checking for SL/SF models," in *Proceedings of the 4th Model-Driven Engineering, Verification and Validation Workshop*, 2007, pp. 33–46.

[28] C. S. Păsăreanu *et al.*, "Model based analysis and test generation for flight software," in *Proceedings of the 3rd IEEE International Conference on Space Mission Challenges for Information Technology*, 2009, pp. 83–90.

[29] M. Satpathy, A. Yeolekar, and S. Ramesh, "Randomized directed testing (REDIRECT) for Simulink/Stateflow models," in *Proceedings of the 8th ACM International Conference on Embedded Software*, 2008, pp. 217–226.

[30] P. Peranandam, S. Raviram, M. Satpathy, A. Yeolekar, A. Gadkari, and S. Ramesh, "An integrated test generation tool for enhanced coverage of Simulink/Stateflow models," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2012, pp. 308–311.

[31] Reactive Systems, "Reactis," Last access: 2013-08-16. [Online]. Available: http://www.reactive-systems.com

[32] T-VEC Technologies, "T-VEC Tester for Simulink and Stateflow," Last access: 2013-08-16. [Online]. Available: http://www.t-vec.com/solutions/simulink.php

[33] The Mathworks, "Simulink Design Verifier," Last access: 2013-08-16. [Online]. Available: http://www.mathworks.de/products/sldesignverifier

Sick But Not Dead Failures - Adaptive Testing, Evaluation and Design Methodologies

Tara Astigarraga[1], Michael Browne[2], Lou Dickens[3], and Ian MacQuarrie[4]
Systems and Technology Group
IBM
[1] Rochester, NY 14626
[2] Poughkeepsie, NY 12601
[3] Tucson, AZ 85744
[4] San Jose, CA 95134
{asti, browne, dickens, imacq}@us.ibm.com

Abstract- **Enterprise data center implementations make significant investments in high availability configurations, redundant hardware, software and Input / Output (I/O) paths that are in many failure scenarios quite successful. However, in spite of all that investment clients are still facing unexpected outages and performance impacts related to a phenomenon referred to as Sick but not Dead (SBND) errors. SBND errors are sometimes lumped together in a category with other related errors including transient errors, partial failure scenarios and soft errors. While SBND errors do have many common characteristics with the errors described above, there are key differences and environment impacts which we will explore further in this paper. We will also present new proactive techniques, inject scenarios and methods to identify, characterize and address SBND failures including cross-component impacts and failures.**

Keywords- Software Testing, Sick but not Dead, Software Engineering, Partial Failure, Transient Error, Soft Failure, SAN Test, Storage Area Network Test, System Test.

I. INTRODUCTION AND MOTIVATION

Despite high availability (HA) configurations, failures are still occurring that are impacting customer environments. Impacts range from varying degrees of performance degradation to complete multi-system outages that often cause enterprise level business outages for extended periods of time. When these types of failures escalate to enterprise level loss of data access events the data verification time for these events can be lengthy and involve numerous business and information technology staff long after the error condition is resolved [1].

The type of failure leading to these impacts is one that exhibits a temporary and reoccurring behavior. Meaning errors are being recovered at various points within the system, however, the errors continue to occur at varying rates. We classify these errors as Sick but not Dead (SBND) failures. These errors are often the hardest failures to identify and can have sporadic but lasting impacts on the environment as a whole. SBND failures currently represent 80% of business impact, but only about 20% of the problems [2].

SBND errors are sometimes lumped together in a category with other related errors including transient errors, partial failure scenarios and soft errors. While SBND errors do have many common characteristics with the errors described above, there are key differences as well. SBND errors by definition derive from a component within the I/O path that is 'sick' meaning behaving in an unorthodox or partially failed fashion but not completely 'dead' or hard failed. Depending on the component exhibiting the SBND characteristics, the symptoms can vary, come and go at different intervals and persist for an extended period before the component eventually reaches a hard fail state or would otherwise persist indefinitely if not for manual intervention. It is this in-between time when the component is defined as SBND.

While there have been examples of the industry trying to address this problem proactively with technologies like IBM's Predictive Failure Analysis and S.M.A.R.T monitoring which has been incorporated in ANSI INCITS T13 Technical Committee, the problems still persist at both the device and system level [3]. Many of the predictive technologies in place today have some obvious constraints. Inherent in a new technology is the lack of experience in being able to correlate performance and calibration data with a reasonable expectation of a failure. It takes technology providers a fair amount of time and maturing of a technology to be able to make reasonable correlations. This increases the opportunity for SBND failures during this maturing time frame. The S.M.A.R.T monitoring standard has a requirement to reset all counters to zero after a firmware modification which may or may not address a component that has or is about to exhibit SBND behavior. The prediction of a failure is in opposition to the economic needs of vendors to not replace components too early under vendor warranty periods. Vendor warranty costs can be increased if the prediction is too opportunistic. These and similar factors and constraints at the component level make it very difficult to design away the SBND failures. At a system level, the holistic environment needs to be able to encounter these conditions with reasonable robustness such that the environment does not degrade to an outage situation. To adequately system test a complex environment SBND errors and scenarios need to be designed and injected into complex environment testing and holistic test observations and evaluations need to be made to determine if the complex environment is robust enough for its intended use.

Complex customer solutions and environments utilizing mixed vendor products and technologies create textbook scenarios for SBND failures to occur. Many products are intolerant of errors from other devices, and although most products respond promptly to hard failure conditions they are much slower to respond to SBND conditions and often do not deploy logic necessary to even detect SBND conditions. With current field solutions, problem determination related to SBND failure scenarios is complex, time consuming and often requires special problem determination lab trace tools and a team of cross-vendor product and solution experts. Current resolutions to SBND failure scenarios are almost always reactive vs. proactive.

One of the common areas for this reactive approach to SBND errors in the I/O path are in the area of multi-pathing function inside or in the I/O code stack for devices. It is typical for a situation to arise in the field that was ultimately caused by a particular SBND failure resulting in an undesirable system level behavior. Almost all multipath software providers have provided documented fixes for these types of failures. An example of a very common multipath driver scenario involving a SBND scenario is having multiple paths across two Storage Area Network (SAN) fabrics where a SBND failure is occurring in one fabric and the multipath software detects a timeout in that fabric. The driver then resends the transaction on another path. The driver then tests the first path finds it available and sends the next transaction down that path which then results in a delayed I/O response which then times out and the multipath software resends the transaction down the good path. When this happens repeatedly the upper level applications like a database start showing severe performance degradations. Fixes were made to the specific Novell SUSE-2011:7794 recommended update with a fix description of 679309: repeated use of flaky paths in multipath module causing performance issue. While this specific example is on a particular product it is the experience of the authors that most multipath software products or functions have had similar fixes generated due to problems encountered in the field. Vendors are working at design time to try an address these situations. For example some vendors have added policy parameters to their multipath software that use a MRU (most recently used) policy to prevent port flapping but eventually have to put the offending path back online if the most recently used path hard fails and then the bad path becomes the most recently used path and the situation can accelerate to a data loss of access event. If the real problem is downstream in an inter-switch link (ISL) the problem will appear to move around making it more difficult to detect correctly and for a multipath software function to behave as needed.

In our system test and SAN labs we have been developing new proactive techniques, protocol inject scenarios and methods to identify, characterize and address SBND failures including cross-component impacts and failures across the I/O path.

A thoughtful and holistic approach is needed in designing these tests. A test engineer can easily create failure scenarios that will never happen or that nothing will be reasonably able to recover from. If the test design does not take these factors into account the test engineer will create scenarios that no development group would agree to develop fixes for. It is therefore critical that test engineers have a very good understanding of multiple components and when things like speed changes occur in the hardware or firmware that may alter the amount of time in either direction for a SBND scenario to occur. Most of the time, these changes and their potential impacts are not explicitly called out in a typical design document. An innocent statement like changed scanning frequency in the firmware to reduce latency could significantly change an error detection rate or behavior somewhere else in the stack of software and firmware.

Our current research related to SBND defects reported shows that the highest number of SBND problems exists along the I/O path. While related problems do occasionally exist within specific internal sever paths they are significantly less frequent, easier to debug and typically contained to a single server and handled via embedded HA mechanism.

Systems generally behave properly when failures are solid or hard failures. It is when components act SBND that system availability is often at risk. In these scenarios failover or recovery mechanisms often do not behave as we should expect them to. Often times the problems are corner cases where they are not easily reproducible and hard to trouble shoot, but continue to plague customer environments. It should also be noted that SBND problems are not something that occur in a particular vendor or product set, but rather a system level event that occurs when one (or more) component(s) in the environment does not always behave consistently. Since the problem does not relate to a particular vendor or component issue it is not a simple fix but rather a system level event that must be fully understood, tested and addressed by all vendors in a distributed systems SAN environment.

The focus of this paper will be on SBND failures related to the I/O path in distributed systems Fibre Channel (FC) SAN and Fibre Channel over Ethernet (FCoE) environments. In this paper we will better define and characterize SBND failures, explain the impacts they can have on complex customer environments and introduce new testing techniques and injections we have deployed in our system test labs. We will also explain the methods used to evaluate the effectiveness of error recovery related to SBND conditions.

II. FAILURE TYPES AND CLASSIFICATIONS [4]

Traditionally, network path failures are viewed as falling into one of two categories, “permanent” and “temporary”.

Perhaps the most well understood and easiest to manage are the permanent failures which result from a catastrophic failure of a network component. These failures are typically persistent failures where all commands routed to the failing path(s) will fail. Commands are recovered through retry down alternate paths and the failing paths are permanently removed from service.

The second category of failure is temporary and transient in nature. These failures can arise from numerous sources including bit flips in electronics due to alpha particle or cosmic rays, as well as electrical noise from intermittent contacts and code defects to name a few. These can produce temporary command failures which are recovered through a single retry operation. These tend to be isolated events, handles via low level recovery and thus most often go completely undetected.

Both of these traditional failure categories are handled well by existing multipath drivers available in the industry today and in fact rarely result in any adverse effects on the operation of the system.

Unfortunately, as the speed and complexity of high speed networks has and continues to increase over time a third category of SBND failures has emerged. These failures are also temporary like the second category; however, in addition to being temporary they are also recurring at varying rates. These SBND failures can arise from marginal components or components and network routes that are insufficiently sized or over subscribed for the volume of network traffic present. Often times these failures are provoked by secondary conditions such as an instantaneous increase in network traffic or a convergence of network traffic. These types of conditions can reduce the quality of a network path(s) resulting in a propensity for them to produce temporary failures. SBND failures are very difficult for the server's multipath driver to detect and typically require an independent monitoring system, therefore, for the driver is impeded from taking action to eradicate the fault and, therefore, the condition to persist for an extended period of time. In many cases the fault persists indefinitely or until manual intervention is performed. SBND failure conditions often drive recursive error recovery by the attached servers which leads to symptoms ranging anywhere from moderate performance degradation to complete system outage.

A further complication of this third category of failure is its difficulty to isolate and resolve. Since commands from servers to storage traverse a large number of switches and links the precise component or conditions responsible for the failures are difficult to identify. Additionally, the underlying problem cannot be contained within the network itself and in most cases the network is not capable producing actionable fault indications that would enable prompt response and resolution from the network administrator.

The existing multipath drivers in use today are not capable of adequately handling this intermittent/recurring failure condition. For the most part all multipath drivers behave in a similar fashion in that they detect and take action on what is seen as individual and disparate events. Path management functions to remove and return paths to service are determined based on the outcome of these individual events.

For example, when a command failure is encountered the recovery action involves a retry operation on the same or alternate path. If one or more subsequent command operations fail on the same path, depending on the thresholds in place, the path will be determined to have failed and the path will be removed from service (first failure category described above is assumed). If subsequent commands are successful the error will often be considered temporary (second failure category described above is assumed) and the path will remain in service. Most multipath software also includes a path reclamation function that periodically tests the availability of each path through the network. If the path test is successful on a path that had previously been removed from service that path will be placed back in service. In response to the intermittent/recurring failure the multipath driver will either leave the failing path in service or remove the failing path from service only to return it to service a short time later following a successful completion of the path test performed by the path reclamation function. A behavior often observed as a result is continuous cycling of paths between offline and online states. It can be seen that based on the application of such logic for both removing and returning paths to service that the implementation of the current multipath drivers are not capable of responding appropriately to SBND failure conditions and, therefore, will not be effective at isolating servers from the negative effects of this condition.

Because specific components and/or conditions associated with these types of SBND failures in the network are often difficult to isolate, the ability to automatically detect and respond to these failures from within the multipath driver is critically important and in fact essential to maintaining a high quality of service.

III. Common Characteristics of SBND Failures

Most SBND failures are not obvious product failures. Often when problem determination begins all individual products in the environment appear 'healthy' and existing internal diagnostics are not reporting any serviceable events. Even error log reviews often come up clean, making problem determination very difficult. SBND problems by definition are transient errors, meaning network component or a product is temporarily misbehaving, making the side-effects or symptoms in an environment often appear and disappear.

SBND failures are frequently first noticed at the application and/or database layer and are most often initially reported by the customer. The tables below lists the most common impact symptoms and characteristics displayed when SBND failures are encountered.

TABLE I. COMMON SBND IMPACT SYMPTOMS

Moderate to severe performance degradation occurring at sporadic intervals or sustained
Transaction queuing and timeouts
Application abends
HA node failovers
Jobs running longer than usual
Mirror or replication times exceeding Service Level Agreements

TABLE II. COMMON SBND CHARACTERISTICS

Not an obvious product failure, individual products in the environment appear 'healthy' even after detailed internal dump analysis at highest levels of product support
Fault tolerance mechanisms not seeing errors and don't react.
Hard for software and monitoring products to detect, internal diagnostics often do not find anything
Symptoms often appear and disappear
Symptoms often amplify over time

Note: the two tables above were compiled using defect data from problems that were encountered in the IBM system test labs and the IBM field support group from 2010 through 2013.

One might fail to realize the size and/or scope of a SBND failure, by examining the symptoms alone. This is because SBND failures commonly create a sympathy sickness throughout the entire network. Sympathy sickness is when a single device or condition in one part of a network impairs the performance of other devices or other parts of the network. The list below details the most common contributors to SBND failures:

A. *Flaky adapter cards and interface modules*

Adapter cards and interface modules often do not hard fail. Instead they degrade over an extended period of time producing 1000's of bit errors in the process.

Thus a single bad SFP or adapter card in an E-port, can affect the performance of 100's or 1000's of initiators that have their frames transported over the inter-switch link (ISL). A recent study by researchers at the University of Illinois at Urbana Champaign and NetApp Inc. suggests that the majority of failures in data storage system/sub-system are not caused by disk failures, but are being caused by link errors [5]. The study asserts that up to 80% of the storage system failures are not in the disks at all. The authors surveyed approximately 39,000 storage systems with 155,000 enclosures containing roughly 1.8 million disks over a period of 44 months. During that period only between 20% and 50% of the disk system failures were due to the disk drives. The rest came from other causes, most notably SAN component interconnection problems [5].

B. *Dirty connections and cables*

Contaminated connectors and/or interface modules introduce bit errors as traffic rates increase. A link may operate with an acceptable bit error rate until such time as it is loaded down. This anomaly makes dirty connections/cables especially difficult to isolate and identify. In an article written by Steve Lytle from JDSU entitled "Fiber Connector Cleanliness Overcoming a 'Dirty Little Secret'," Steve says, "More than 75 percent of troubleshooting in optical networks results from dirty fiber connectors, a stunning fact first learned by high data rate equipment manufacturers, and later by transport installation teams in the telecom sector. Many in the cable industry may find this surprising, but the problem exists and is quickly becoming intolerable as fiber networks expand" [6].

In 1990 equipment manufacturers were experiencing a plague of dirty fiber connectors, which lead to the establishment of a team of industry experts who performed practical research within a group called iNEMI. This research is now one pillar of a pending international standard that prescribes inspection procedures and pass/fail criteria for manufacturers and operators of fiber-optic networks (IEC-61300-3-35) [7].

Figures 1 and 2 show the light loss and back reflections that occur when there is contamination in a connector. These two figures were created by iNEMI as part of their investigation.

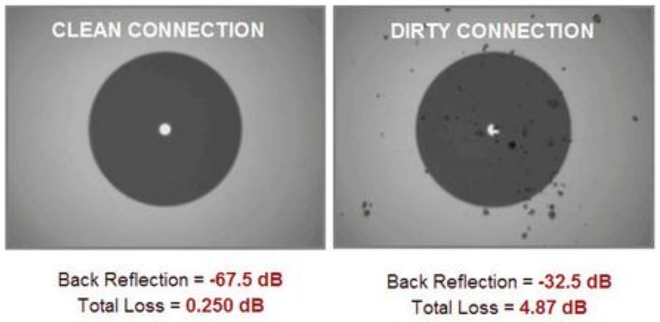


Figure 1. How contamination affect light loss [7]

When contamination is present light levels can be dramatically reduced, as seen in Figure 1. Contamination produces two undesirable side effects, 1) loss of light, 2) reflections. The loss of light reduces the distance that two ports can reliable communicate. Back reflections cause optical resonance in the laser, which creates optical noise further reducing the distance for reliable communication.

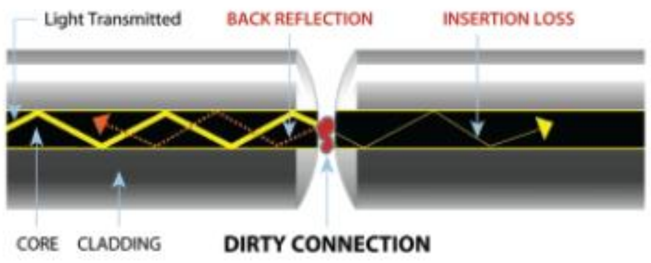
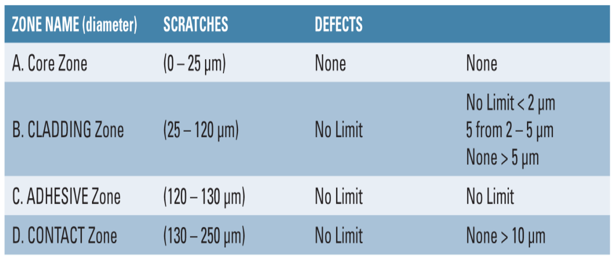


Figure 2. How contamination affects light signals [7]

Cisco, a leading network equipment manufacture, also recognizes that contamination is a problem; they created an Inspection and Cleaning Procedures for Fiber-Optic Connections document in which they state "Any contamination in the fiber connection can cause failure of the component or failure of the whole system. Even microscopic dust particles can cause a variety of problems for optical connections. A particle that partially or completely blocks the core generates strong back reflections, which can cause instability in the laser system. Dust particles trapped between two fiber faces can scratch the glass surfaces. Even if a particle is only situated on the cladding or the edge of the endface, it can cause an air gap or misalignment between the fiber cores which significantly degrades the optical signal.

- A 1-micrometer dust particle on a single-mode core can block up to 1% of the light (a 0.05dB loss).
- A 9-micrometer speck is still too small to see without a microscope, but it can completely block the fiber core. These contaminants can be more difficult to remove than dust particles" [8].

Figure 3 shows JDSU's recommendation for acceptable levels of contamination, which is based on iNMEMI investigation and years of experience in the test and measurement world.



ZONE NAME (diameter)	SCRATCHES	DEFECTS	
A. Core Zone	(0 – 25 µm)	None	None
B. CLADDING Zone	(25 – 120 µm)	No Limit	No Limit < 2 µm 5 from 2 – 5 µm None > 5 µm
C. ADHESIVE Zone	(120 – 130 µm)	No Limit	No Limit
D. CONTACT Zone	(130 – 250 µm)	No Limit	None > 10 µm

Figure 3. JDSU's Recommendation for Contamination [6]

C. *Temporally exceeding the capacity limits of a port/device*

Part of a SAN administrator's job is to insure that network capacity is never exceeded. Normally this is not a problem; ports and/or devices that are communally operating at or near their capacity are easily identified. Transient loads however are different, they can only be detected while they are occurring, which makes them very elusive.

When a port/device reaches its capacity one of two things occur, 1) frames are dropped or, 2) frames are buffered until such time as they can be delivered, which can produce undesirable side effects including latency and bottlenecks.

When frames are dropped or discarded in the network it often results in damaged SCSI exchanges which need to be timed out, and subsequently aborted and re-driven by the host device driver. This type of error recovery is very costly given the read/write timeout interval for SCSI retry is in the 20 to 60 second range depending on operating system. Error recovery for command timeouts is generally tolerated at the application and database layers as long as they are transient events; however, recursive recovery for command time-outs is rarely sustainable even when occurring at what seem to be relatively low rates.

Latency and bottlenecks both create back pressure on other devices throughout the SAN. These devices are forced to wait until the condition is resolved before they can resume sending and/or receiving frames. If the wait is long enough (typically 500ms) the switch will begin discarding frames in an attempt to limit the scope of the impact and command time-out recovery will be required on all damaged SCSI exchanges.

D. *Buffer to buffer credit problems*

A port must have buffer-to-buffer credit in order to originate a frame. Ports without buffer-to-buffer credit are forced to wait until such time as they receive a credit, which creates latency, bottlenecks, and/or opportunity for time-out conditions. There are a number of different circumstances that can lead to buffer-to-buffer credit shortages, these circumstances are listed below:

1) *Long Links or High Latency Links:*

All ports start out with an initial buffer-to-buffer credit that is established during the login process. The default initial buffer-to-buffer credit value is a median value that is adequate under normal conditions. However, if the links are long or the link has a high latency then the default initial buffer-to-buffer credit values will not be adequate for this environment.

2) *Lossy Links:*

When a link is experiencing errors it corrupts the traffic flowing over it (frames and primitives). When a Receiver Ready (R_Rdy) or a Virtual Circuit Ready (VC_Rdy) is corrupted in flight they are discarded by the recipient, which result in a loss of credit. Over time these credit losses can slow a link to a crawl and severely impact its performance.

3) *Low Speed Device in a Critical position:*

When higher speed devices are communicating with lower speed devices buffer-to-buffer credit is used to throttle the frame origination of the high speed device. Normally, networks are configured such that low speed devices are not in critical paths. However, in the event of a link failure a device could be placed in a critical path by a routing protocol such as Fabric Shortest Path First (FPFS).

E. *Compatibility issues*

All Fibre Channel equipment vendors have a support matrix where they document tested and supported configurations. Most vendors do a good job of keeping this information current within the first several years of a products life cycle, however, in time the information becomes incomplete and even misleading, which can result in interoperability issues that can negatively impact performance and/or availability. Additionally, vendor provided migration paths from aging legacy hardware to newer offerings can also introduce some risk. Many of the methods and functions used to provide a migration path such as "switch interoperability mode" do not receive the same level of testing across the full range of configurations and conditions as best of bread environments and therefore are at higher risk of encountering defects. Moreover, configurations intended primarily to be used to facilitate migrations often end up being a permanent part of the environment which can further compromise availability.

IV. Emerging Factors

The incident rate for the SBND category of conditions has been on the rise since SANs were initially introduced in the early 1990s. This has occurred as a result of a multitude of influencing factors: (1) Incremental increases in the speed of network ports and devices have placed a higher demand on the quality of the transport layer. Higher speed networks have a higher sensitivity to degradations in link and device quality which can result in transmission failures. (2) Incremental increases in server capacity to drive IOPs made possible by advancements in processor and bus speeds increases the potential to drive port and device utilization beyond their reliable limits. (3) Increases in port and device utilization have also occurred due to increases in I/O density brought about by the increasing use of virtualization of both servers and storage. (4) Added complexity of SAN topologies associated to speed matching required for maintaining legacy hardware, multi-site replication architectures, as well as the sheer scale of networks resulting from growth.

Additionally, the impact rate associated with SBND conditions has also been on the rise due in large part to the increased demand on transaction based workloads. In the past, degradation in performance caused by SBND could often be tolerated to some extent while the problem was being diagnosed and resolved. This is rarely the case any longer where any degradation in performance impeding the system's ability to sustain transaction volume and quality of service are likely have negative consequences for the business and will therefore be considered an outage.

As we move to new computing models such as integrated infrastructure, software defined, and cloud it will become even more imperative that these SBND conditions are dealt with quickly and autonomously. Enabling system design and development to meet these objectives requires a testing approach and methodology that allows these systems to be measured and evaluated on how they perform when subjected to SBND conditions.

V. Test Approach

In a proactive attempt to better address and improve test and design around SBND customer failures, IBM introduced an internal quality improvement effort to better define, categorize and test SBND failures. As part of this ongoing effort, the IBM Systems and Technology Group labs have started introducing a variety of SBND symptoms into complex system test environments using a four-pronged approach. 1. Build a center of competency around identifying, characterizing and debugging SBND failures in the I/O path. 2. Target modified reliability, availability and serviceability (RAS) microcode to better identify and flag SBND failures for troubled areas. 3. Targeted test case coverage related to SBND failures, symptoms and characteristics. 4. Redefine the criteria for success and failure of the new test cases to better reflect the symptoms impacting data centers today. It is no longer sufficient to measure success of error recovery based solely on permanent vs. recoverable error conditions. It is common with SBND conditions for service interruptions to occur yet all I/O be recovered at various layers of the I/O stack, therefore, it is clear that additional criteria that incorporates performance attributes is required to effectively test for this condition. Recursive error recovery while it may be successful has a significant impact on throughput. Although error recovery will always have a measurable impact on performance, error recovery needs to be able to not only recover I/O but also determine and execute actions necessary to remove failing resources/paths from the I/O path. The speed and accuracy at which error recovery is able to accomplish this is the critical measurement of success in handing SBND conditions.

This paper will focus on the 3rd and 4th prongs described above as they relate to increased SBND testing and early results.

A. *Targeted test case coverage related to SBND failures, symptoms and characteristics*

In late 2010 the SAN test labs within IBM began technical analysis on SBND errors and targeted ways to not only inject SBND failures, but to proactively monitor the environment as a whole for related defects and outages. This was a detailed and controlled approach consisting of injects in three primary locations within the I/O path, as outlined in Figure 4 below.

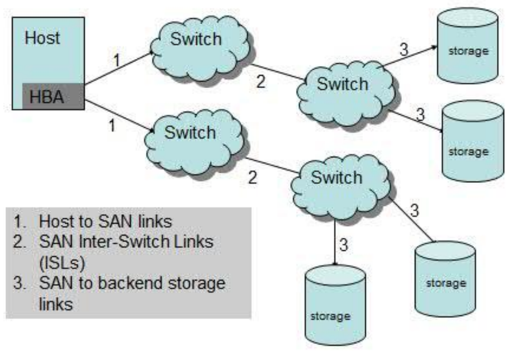


Figure 4. SAN Inject Points

Once the inject areas were established and test tools in place we began targeted testing covering the most frequent SBND symptoms and characteristics described in Tables I and II. Table III below outlines some of the test injects symptoms and test case examples that were created to inject SBND symptoms into our SAN environments to monitor for proper handling and unintended side effects across the environment.

TABLE III. SBND TEST SCENARIO INJECTS

Symptom:	*Types of Injects Used:*	*Test Case Examples:*

Severe Performance Degradation	1. Credit starvation 2. Inject Delay	1. Replace R_Rdy primitives with IDLE/ARB (FC), inject PFCs for Class 3 traffic (FCoE). 2. Hold all frames for *x* microseconds
Mirror or Replication times exceed Service Level Agreement	1. port flaps 2. drop frames 3. jitter	1. Port shut/no shut activity (FC,VFC,Eth) 2. Drop every *xth* frame in each direction 3. Corrupt sof, eof, crc and other header data
I/O redrives or near redrives	1. drop, corrupt or re-order data frames 2. short holds of frames	1. Target data frames and drop or re-order 2. Hold all data frames and/or transfer ready frames for *x* seconds.
Application sensitivity to Recoverable I/O Events	1. virtual link jams 2. link resets 3. corrupt frames	1. FDISC drops, VFC jitter, VSAN jams 2. Inject NOS, OLS, LR and/or LRR onto link 3. Corrupt bits in the FC or FCoE header and recalculate CRC
Product behaviors related to unforeseen external trigger events	1. protocol violations 2. unexpected data returns 3. partial recovery scenarios	1. Inject protocol deviations from standard and monitor destination handling 2. Return Check Cond to write exchange 3. Drop data frame, then drop subsequent ABTS, allow re-driven ABTS to flow through un-jammed.

B. *Redefine the criteria for success and failure of SBND test cases to better reflect the symptoms impacting data centers today*

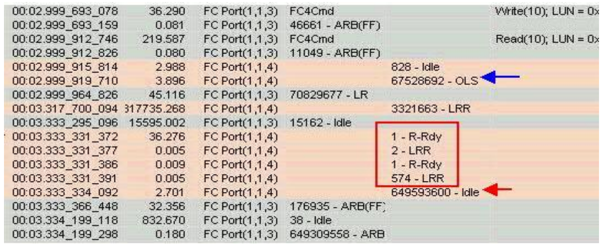
For years the industry has measured the success/failure of error recovery test cases using *permanent I/O error* as the measurement. In today's client environments the majority of SAN impacts events are occurring as a result of the performance degradation associated with the error recovery, they are not typically caused by permanent I/O errors. Accordingly, verifying that SCSI and related error inject scenarios recovered and did not result in a permanent I/O error is no longer sufficient.

In today's high speed environments the time it took to fully recover and the impact on other I/O (often times non-related I/O) is essential. As Server consolidation and virtualization trends continue, the impacts of non-permanent I/O errors will continue to plague environments and cause severe performance impacts until we change the way we test, develop and measure recovery.

In order to evaluate the effectiveness of error recovery for SBND conditions we had to redefine the test criteria around performance attributes. The critical attributes measured and quantified are 1) the amount of degradation that occurs during the recovery and 2) the length of the recovery defined by the time it takes for performance to return to nominal.

VI. RESULTS

Overall, we established a test suite consisting of over 100 unique SBND test cases, which are run in a controlled SAN environment allowing us the capabilities to inject a single error (or bucket combinations of errors) and monitor the environment as a whole. The majority of the problems we have identified are defects that would have been near impossible to detect and correlate in a customer environment. The ability to understand which variables are being injected at which time and location in the SAN and watching all associated host, switch and storage logs provides the ability to correlate and connect events that otherwise would have appeared to be non-related. Further, having packet level traces at each point in the SAN allows the ability to deep-dive into the traces. Figure 5 below illustrates one SBND inject example where every 5 minutes the Not Operational primitive sequence (NOS) was injected to simulate a bouncing or partially failed port in the SAN. Figure 5 below shows the subsequent behaviors following one of the NOS injects which resulted in failed link initialization. For link initialization to complete successfully following our NOS injects the primitive sequences OLS/LR/LRR/IDLE/IDLE have to be traded sequentially. In Figure 5, we can see one SAN vendor sent extra R_RDY primitives and LRRs prior to sending the final IDLE packets required to complete link initialization. These extra packets prevented proper link initialization and resulted in a substantial delay, impacting link recovery by 20 seconds. Delays of this magnitude produce excessive service times as seen by the application and may result in transaction timeouts, as well as potentially expose application layer sensitivities which could lead to an outage. After the SBND defect was fixed and verified the recovery time dropped from 20 seconds to just milliseconds.



```
00:02.999_693_078      36.290  FC Port(1,1,3)  FC4Cmd                                        Write(10); LUN = 0x
00:02.999_693_159       0.081  FC Port(1,1,3)  46661 - ARB(FF)
00:02.999_912_746     219.587  FC Port(1,1,3)  FC4Cmd                                        Read(10); LUN = 0x
00:02.999_912_826       0.080  FC Port(1,1,3)  11049 - ARB(FF)
00:02.999_915_814       2.988  FC Port(1,1,4)                   828 - Idle
00:02.999_919_710       3.896  FC Port(1,1,4)                   67528692 - OLS
00:02.999_964_826      45.116  FC Port(1,1,3)  70829677 - LR
00:03.317_700_094 317735.268  FC Port(1,1,4)                   3321663 - LRR
00:03.333_295_096  15595.002  FC Port(1,1,3)  15162 - Idle
00:03.333_331_372      36.276  FC Port(1,1,4)                   1 - R-Rdy
00:03.333_331_377       0.005  FC Port(1,1,4)                   2 - LRR
00:03.333_331_386       0.009  FC Port(1,1,4)                   1 - R-Rdy
00:03.333_331_391       0.005  FC Port(1,1,4)                   574 - LRR
00:03.333_334_092       2.701  FC Port(1,1,4)                   649593600 - Idle
00:03.333_366_448      32.356  FC Port(1,1,3)  176935 - ARB(FF)
00:03.334_199_118     832.670  FC Port(1,1,3)  38 - Idle
00:03.334_199_298       0.180  FC Port(1,1,3)  649309558 - ARB
```

Figure 5. Protocol Trace Review

The protocol trace analysis and frame level debug functionality provides enhanced problem determination capabilities, that when combined with associated host, switch and storage logs present a clear picture of the problem and greatly assists with cross-vendor problem determination.

Typical product system test environments and test plans are designed to analyze and validate recovery capabilities in a product or system offering along with potential implementation architectures and then inject hard errors to determine if products under test were behaving according to specification and customer requirements. A high level example would be a system test environment that had been designed and implemented with full redundancy of all components in order to minimize Service Level Agreement (SLA) violations [10]. The test engineer would then introduce failures of the components at injectable points in the configuration to validate and verify the system offering would meet SLA requirements. What this technique misses is the "almost errors" that are not specified or articulated as customer requirements. Additionally, there is some level of subjectivity to a SBND event actually occurring and convincing the designers that such a situation would or could exist in the real world. A test engineer also has to use reasonable judgment in designing the injection as any SBND injection can be pushed to unrealistic limits and then the test can be declared invalid. For example, when testing credit starvation one must be cautious in the rate of R_Rdy (frame buffer credit) drops that are injected as too many will cause link resets, replenishing credits back to the agreed upon limit during login. For SBND scenarios, the tester would want to identify the buffer credits allotted during login and drop R_Rdys at a rate which slowly impacts the environment without causing an immediate link reset. It is this careful balance that must be pursued in the test design and execution. Having a test engineering center of competency for SBND problems that can provide real world patterns of these injections is critical to wining the subjective discussions between test engineers and designers.

Since starting this work in 2010 we have seen a dramatic spike in internally found SBND related defects being identified and fixed in system test. In 2010 when we started this testing only 5% of the defects found in SAN system test were related to SBND error handling. In 2012-2013 SBND related defects represent 52% of the overall defects opened by the SAN system test teams. The defects opened are spread across multiple vendors and I/O path components including operating systems, host HBA/CNA firmware and drivers, multipath drivers, SAN and FCoE switch code and storage firmware and drivers. Although defect signatures are often unique, there are common trends that emerge; we will examine those trends in more detail below.

A. Common SBND defect trends

1) Bit errors from a simulated SBND device often lead to error recovery escalation

Bit errors from a simulated SBND device often lead to device driver error recovery escalation and have unintended impacts on non-related I/O streams. Fibre Channel standards allow for a single bit error to occur only once in a million bits (1 in 10^{12}) [10]. In a real world example at 16Gbs that would allow for 56 bit errors per hour. On healthy links bit error occurrences are typically much less frequent, but in any environment occasional bit errors will occur and SBND testing helps to confirm proper handling and recovery. SBND testing related to bit errors typically consists of corrupting or flipping bits at a given location in the environment at a predetermined rate. Inject rates range from multiple errors per second to one error per hour. The frequencies at which the errors are injected greatly vary the results and handling practices. When SBND errors are hit by the same host within a given period of time (varies by host adapter vendor and multipathing option) the host device often reacts by escalating recovery methods and selecting more aggressive task management and error recovery options. The SCSI protocol has a number of task management function defined, some of which impact single initiators tasks and others that affect all commands in the task set or even all commands running within a SCSI target [11]

For example, in one scenario two errors encountered on the same path within 10 seconds led the host device to issue a SCSI Logical Unit (LUN) Reset to the associated storage target LUN. A LUN reset will abort all in-flight commands for the associated LUN and take the LUN back to power on state.

SCSI Target Reset is another task management option which is even more severe than the LUN Reset given its broader scope. A Target Reset will abort all in-flight commands for all LUNs within a given storage target and then take them all back to power on state. It is typically used after other forms of error recovery failed or during frequent SBND events. Target reset was obsoleted by the standard in 2002 due to the harsh impacts it has on environments and non-related I/O streams, however, it is still in active use by multiple adapter vendors today [11].

In our SBND test environments we give special attention to target resets and review each scenario in which they were issued. The goal is to reduce the use of target reset when a different form of task management could be used. In a SBND environment if one host side link is plagued with errors and that host aggressively uses task management commands including Target Reset all other host devices zoned and setup on the shared storage target will be continually impacted. This will result in a large scale performance impact and a difficult to isolate SBND environment impact condition. In the field these problems are challenging to debug as often symptoms first show up on non-related hosts and it takes technical experts and combing through host, switch and storage logs to ultimately identify a single rogue host device impacting the overall environment.

Ideally, in the case of a host device that has a link plagued with SBND errors the multipath driver would identify the problem and remove the paths from their active selection lists, thereby preventing further errors and potential impact to non-related hosts and devices.

In the past year advancements have been made in multipath handling of SBND errors as a result of SBND testing, design reviews and client impact events related to SBND errors. Several multipath drivers are becoming more aggressive at failing paths and more cautious to continually return those same paths to service.

2) Error injects on ISLs are harder to detect by multipath drivers, typically paths are quickly taken offline then put back online within a few seconds

Many of the classic SBND symptoms are caused by a partially failed or flaky ISL. Intermittent problems are difficult to assess, ultimately the fabric must determine if a problem is critical enough to disable the ISL port [12]. Buffer credit depletion, link flapping or marginal cable or SFP components are most often the cause of SBND symptoms on ISLs. These symptoms are often detected by frame discards, offline/online events or cyclic redundancy check (CRC) errors on ISL switch ports. SBND ISLs can cause intermittent I/O failures, application layer timeouts and severe performance degradation.

To understand the impacts on the environment lets first explore how multipath health checkers typically work. Host multipath software characteristically uses Test Unit Ready commands to check health status of a path. After one or two successful Test Unit Ready commands with good status the path is assumed to be healthy and brought back online if previously in an offline state. In the case of SBND ISLs the errors are intermittent and multipath has no means of correlating the errors across time and detecting the path as SBND.

The end result of SBND ISLs combined with a multipath driver not equipped to handle SBND failures is we see I/O exchanges fail and the associated paths are marked as failed, however, shortly after (typically around 2 seconds) the paths are brought back online. The disproportionately high number of SCSI I/O commands versus Test Unit Ready commands makes it likely that an I/O command will fail and a link maintenance command will succeed. This leads to the offline/online cycling of paths which could continue until manual intervention or the path degrades to a hard fail.

3) Virtualization Components have higher defect rates related to SBND.

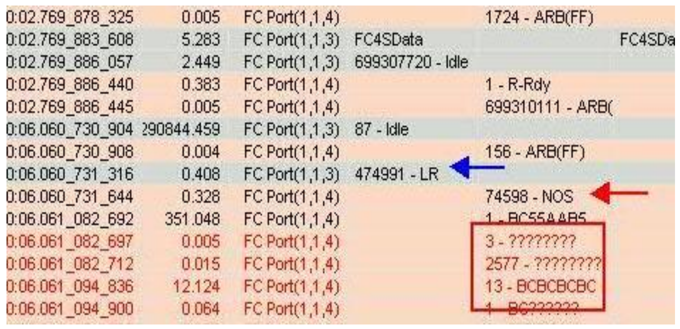
Virtualization technologies provide many efficiencies and capabilities and continue to advance and mature. However, virtualization also creates new challenges and debug scenarios for network administrators and technology vendors. The consolidation of virtual servers and virtual links adds complexity and can create bottlenecks, sporadic load patterns and makes it harder to troubleshoot and identify SBND components in a complex virtualized environment.

Links shared across multiple initiators are harder to debug and the sporadic nature of SBND events coupled with virtualized environments makes these errors difficult for multipath drivers to detect and respond appropriately to. Additionally, some server virtualization technologies separate link layer and SCSI layer management responsibilities into entities that reside on different virtual images. This separation adds additional complexity and makes it more difficult for the firmware, driver stack(s) and multipathing software to stay in-sync and respond properly to SBND events.

Host side virtualization related SBND defects often come down to the host adapter firmware detecting SBND events but not passing detailed information up the stack to the SCSI emulation layer so that the device drivers and multipath drivers can make informed pathing decisions. The same concept also holds true for the storage virtualization layer, SBND defects often result from communication breakdowns and faulty state management synchronization between the virtualization layer and backend storage controller(s).

4) Buffer to Buffer Credit Recovery

Buffer to buffer credit recovery is basic and key functionality in a SAN environment. SBND Credit related defects typically fell into two categories; devices who did not handle loss of credit(s) properly and devices that could not perform credit recovery properly. For devices that did not handle the loss of credit(s) properly typical symptoms included large performance impacts or flat lines following the loss of a single or a few credits. For devices that could not perform credit recovery properly the typical behavior was often a credit reset attempt that did not complete properly and would lead to a link reset. Fig. 6 provides one example of a target device that could not properly perform credit recovery. Essentially, anytime the target device port or the connected switch port ran out of credit it ultimately resulted in a temporary loss of access event simultaneously for every host in the fabric zoned to that target port. Due to the devices inability to properly handle credit resets the port would fail, loose-sync, then re-initialize and log back in with the fabric. From the initiator side, paths to that storage port would fail, multipathing would fail over and all open exchanges would have to be recovered and redriven. Once the target device was back online and logged in with the switch the hosts would re-login with the target device and lost paths would recover. Had the initial credit recovery attempt worked the credit reset would have been seamless and unnoticeable to the related hosts.


```
0:02.769_878_325      0.005  FC Port(1,1,4)                        1724 - ARB(FF)
0:02.769_883_608      5.283  FC Port(1,1,3)  FC4SData                                 FC4SDa
0:02.769_886_057      2.449  FC Port(1,1,3)  699307720 - Idle
0:02.769_886_440      0.383  FC Port(1,1,4)                        1 - R-Rdy
0:02.769_886_445      0.005  FC Port(1,1,4)                        699310111 - ARB(
0:06.060_730_904 290844.459  FC Port(1,1,3)  87 - Idle
0:06.060_730_908      0.004  FC Port(1,1,4)                        156 - ARB(FF)
0:06.060_731_316      0.408  FC Port(1,1,3)  474991 - LR
0:06.060_731_644      0.328  FC Port(1,1,4)                        74598 - NOS
0:06.061_082_692    351.048  FC Port(1,1,4)                        1 - BC55AAB5
0:06.061_082_697      0.005  FC Port(1,1,4)                        3 - ????????
0:06.061_082_712      0.015  FC Port(1,1,4)                        2577 - ????????
0:06.061_094_836     12.124  FC Port(1,1,4)                        13 - BCBCBCBC
0:06.061_094_900      0.064  FC Port(1,1,4)                        1 - BC??????
```
Figure 6: Credit Reset Causes Link Reset Trace

B. *Closed Loop Process for Field Problems*

IBM Systems test engineers are often brought in to help debug, recreate, analyze or test fixes for client field problems. Many test engineers are also assigned as customer advocates to client accounts that match their industry and technical areas of expertise. These client interactions help the test organizations form tighter client relationships and provide key field data to be shared with IBM test and development labs via closed-loop processes designed to improve test coverage and client experiences. These models help us to improve test coverage, analyze test escapes and perform coverage analysis across the numerous IBM System Test labs worldwide.

As SBND errors are gaining recognition and focus our client relations help us to better understand the impacts these events have on various network design layouts.

Understanding the implications of these errors on client environments help us to test and architect future solutions that will better handle and coordinate SBND events, ultimately reducing the impacts they will have on future environments.

Outlined below is one example of SBND handling enhancements that were submitted as a result of the closed loop process analysis after client impact events.

1) *Multipath Maintenance Improvements*

A large client experienced a storm of command timeouts which escalated to LUN and Target Resets failure conditions resulting in serious performance degradation and application failures. The symptoms stemmed from a SBND ISL that was toggling up/down. The AIX MPIO and SDDPCM multipath software would fail when the ISL path went down and recover after health check commands succeeded on these paths, thus continually failing and recovering the problematic paths. After debug involvement and technical review AIX MPIO and SDDPCM multipath enhancements were made to coordinate failures across time to better detect and handle SBND errors and prevent the perpetual failure and rapid recovery storms of SBND related paths. Starting with SDDPCM v2.6.3.0 a new feature and timeout_policy device attribute of disable_path was introduced. The disable_path attribute will permanently disable a path (not the last path) if it experiences timeouts above the set threshold within a given period of time. The path will stay in the disabled state, until the user manually recovers it [13].

C. SBND Implications on Design Review Process

Traditional design reviews focused on network path failures and error recovery based on two traditional category of events; permanent and temporary. Permanent failures are typically the easiest failures to design for and have solid coverage in design reviews. Temporary or transient errors historically have good design review coverage as well but the number of possible transient errors is much greater than permanent failures, making full coverage more challenging to review than permanent failures.

SBND errors have only recently been added to design reviews. The nature and unpredictability of SBND failures makes design reviews for SBND a constant challenge. As we continue to further understand SBND errors and their implications on complex environments we often find that a fix for one situation sometimes further aggravates another. There is a definite balance to be considered when determining how aggressive we can be when addressing SBND conditions and still ensure implemented solutions benefit all environments.

Recent SBND errors have also uncovered additional temporary or transient error coverage review requirements. Design and code review processes have been updated to review for any single try events that may exist in a code path. A single try event mixed with SBND errors or even a single transient error impacting the frame could result in an unexpected failure.

For example, in our SBND testing we have corrupted fabric login (FLOGI) and fabric discovery (FDISC) extended link services frames, many devices were able to handle these corruptions and re-drive login processes, however, we also discovered initiator and storage targets that were unable to recover from these injects. The implications of this testing was if SBND errors impacted extended link service login frames the devices would not be able to recover and would not complete fabric login. These SBND defects were opened with the product vendors and SBND impacts on extended link services were also added to the closed loop design review process.

VII. Conclusion and Future Work

As complexity, virtualization, business criticality and mixed vendor solutions continue to grow in the IT industry and customer solutions, the need for highly-skilled SBND low-level testing will also continue to increase. In an industry where quality is expected and customer defects can cause costly outages it is no longer sufficient to test products for correct recovery only in hard failure scenarios. We need to continue to put increased focus on solution testing, and further on solution injects and handling of hard failures and SBND failures on any component within the environment. We also need to reevaluate our test pass criteria and design points for complex SAN environments and update them to not only evaluate and design for recovery, but to also evaluate the impact SBND error(s) have on the environment and performance. We need to continue to focus on solutions that can rapidly detect, isolate and address SBND components in an environment. Good progress has been made since we first started this focus in 2010; however, there is still considerable work to be done.

As we continue to expand SBND testing scope described in this paper, we are concurrently pursuing plans to continue this effort with a second phase targeting new inject methods and focus on spreading SBND test capabilities and awareness across IBM test and partner test labs worldwide. Given the economic costs of the tools to inject SBND scenarios and the skill required we are also innovating in economically scalable methods to do this type of testing in more diverse testing and test skill environments. We also continue to drive a close-loop feedback process between IBM test, development and support teams and across OEM partners, ensuring that the SBND defects that have been found are fixed and lessons learned are applied to future product development and monitoring capabilities.

It is our hope and vision that impacts of SBND failures be understood across the industry and that more SBND testing and proactive measures are taken to help minimize the impacts these failures have on the environments of the future.

VIII. Acknowledgments

The authors would like to thank their employer, International Business Machines (IBM) for supporting their efforts to produce educational content. We would also like to thank those parties who provided quotations and background art for use in this paper.

References

[1] Tara Astigarraga, Michael Browne, and Lou Dickens "Sick But Not Dead Testing – A New Approach to System Test", VALID 2012, ISBN: 978-1-61208-233-2 http://www.thinkmind.org/index.php?view=article&articleid=valid_2012_1_30_40098 (last accessed 12/09/2013)

[2] A. Hanemann, D. Schmitz, and M. Sailer "A framework for failure impact analysis and recovery with respect to service level agreements", Services Computing, 2005 IEEE International Conference, Content resides on, vol. 2, no., pp. 49- 56 vol. 2, 11-15 July 2005 doi: 10.1109/SCC.2005.10 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1524423&isnumber=32587 (last accessed 12/09/2013)

[3] ATA8-ACS, ANSI Draft Standard T13 Project 1699D, 2004. http://www.t13.org/documents/UploadedDocuments/docs2007/D1699r4a-ATA8-ACS.pdf (last accessed 12/09/2013)

[4] Ian MacQuarrie, William Carlson, Jim O'Connor, Limei Shaw and Shawn Wright, "Configuring SDDPCM for High Availability", IBM Redpaper, October 2012. http://www.redbooks.ibm.com/redpapers/pdfs/redp4928.pdf (last accessed 12/09/2013)

[5] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky "Are Disks the Dominant Contributor for Storage Failures – A Comprehensive Study of Storage Subsystem Failure Characteristics", USENIX Conference, Storage Technologies 2008 (FAST'08), https://www.usenix.org/legacy/event/fast08/tech/full_papers/jiang/jiang_html/paper.html (last accessed 12/09/2013)

[6] Steve Lytle, "Fiber Connector Cleanliness – Overcoming a Dirty Little Secret", October 2008. http://www.cablefax.com/tech/operations/bestpractices/31826.html (last accessed 12/09/2013).

[7] IEC-61300-3-35, IEC Standard, 2009, avalable for purchase at http://kandk-fi-bin.directo.fi/@Bin/f1c7665a67b79ef92a25406cc614ff18/1386673907/application/pdf/101256/IEC_61300-3-35_Standard.pdf (last accessed 12/09/2013)

[8] Cisco Corporation, "Inspection and Cleaning Procedures for Fiber-Optic Connections", September, 2006. http://www.cisco.com/en/US/tech/tk482/tk876/technologies_white_paper09186a0080254eba.shtml (last accessed 12/09/2013),

[9] B. Rogers, "z/OS 1.11 Sysprog Goody Bag", SHARE Session 2228, March 2010. http://mobile.share.org/client_files/SHARE_in Seattle/S2228RR092920.pdf (last accessed 12/09/2013).

[10] Jon Tate, Pall Beck, Hector Hugo Ibarra, Shanmuganathan Kumaravel and Libor Miklas "Introduction to Storage Area Networks and System Networking", November 2012, ISBN 0738437131. http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf (last accessed 12/09/2013).

[11] Lou Dickens, "Fibre Channel for the 21st Century" February, 2013, ISBN 978-1-939883-00-1 pp. 219-223 www.yahweheducation.com] (last accessed 12/09/2013).

[12] Brocade Corporation, "SAN Fabric Resiliency Best Practice v2.0", July 2013. http://www.brocade.com/downloads/documents/best_practice_guides/san-fabric-resiliency-bp.pdf (last accessed 12/09/2013).

[13] IBM Multipath Subsystem Device Driver Path Control Module,(PCM) Version 2.6.3.0 Readme for AIX, April 2013. ftp://ftp.software.ibm.com/storage/subsystem/aix/2.6.3.0/sddpcm.readme.2.6.3.0.txt (last accessed 12/09/2013).

[14] FC-FS-3, ANSI Standard 5.2.4-5.2.5, 2008. http://www.t11.org/t11/stat.nsf/1158203694fa939f852566dc0049e810/a7bf7ee8c25bd7b9852572e50079eed4?OpenDocument (last accessed 12/09/2013).

[15] FC-MI, ANSI Standard 3.2.14-3.2.34, 2001. http://www.t11.org/t11/stat.nsf/1158203694fa939f852566dc0049e810/86a95105bd279d148525772000567d1d?OpenDocument (last accessed 12/09/2013).

Towards Evolvable State Machines and their Applications

Dirk van der Linden[1], Wim Ploegaerts[2], Georg Neugschwandtner[1], and Herwig Mannaert[1]
[1]University of Antwerp, Belgium
{dirk.vanderlinden, georg.neugschwandtner, herwig.mannaert}@uantwerpen.be
[2]PWCS bvba, Belgium,
wim.ploegaerts@pwcs.eu

***Abstract*—Since several decades, the pressure on organizations to swiftly adapt to their environment has been increasing. At the same time, the complexity of products and services has been growing. One of the consequences is the increasing importance of the evolvability of software, whether this software supports production control systems in industry or business information systems. Over the past decades, finite state machines have become an increasingly popular tool for modelling behavioural aspects of software. This paper presents an explorative attempt to define design rules and constraints that should be applied to state machines to enable evolvability. Our design of an evolvable state machine is based on Normalized Systems Theory. This design is discussed in the context of automation systems as well as more general information processing applications.**

***Keywords*-Normalized Systems; Evolvability; Finite State Machines; Automation Systems; Information Technology.**

I. Introduction

Current organizations need to be able to cope with increasing change and increasing complexity in most of their aspects and dimensions [1]. We shall call a system *evolving* when changes in terms of the system's capabilities occur. The effort or cost required for adding or changing a specific capability is a property of a system – the property of *evolvability*. Evolvability is increasingly important for organizations to allow them to swiftly adapt to an agile and complex environment. The evolvability of production control and information systems is the primary focus of this paper. We will present a design for evolvable state machines, based on Normalized Systems Theory (NST). Its concepts can be applied to all state machines; examples will be discussed for control systems, kiosk software and business process automation.

Evolvability is a critical non-functional requirement on software. Better evolvability favourably impacts the challenging task of software maintenance, where *adding a six-lane automobile expressway to a railroad bridge is considered maintenance* [2]. In their review of evolvability as a characteristic of software architectures, Ciraci and van den Broek [3] define it as "a system's ability to survive changes in its environment, requirements and implementation technologies." However, evolvability is hard to measure, and existing software development methodologies focus on functional requirements almost exclusively.

Maintenance activities often disrupt normal (productive) system operation. If dynamic reconfiguration can be achieved, downtimes of systems can be reduced: a change which can be performed without a complete shutdown is called a 'dynamic reconfiguration' – in contrast to a 'static reconfiguration', which requires the complete shutdown of a system [4]. The ability of an evolving system to introduce a change by dynamic reconfiguration is a special property of the system and the type of change.

Having to stop production for maintenance can be especially costly for continuous production systems or 24/7 production operations. Consequently, production loss or delay of production due to an update of the system must be considered an indirect maintenance cost. In fact, there is a similar cost in information systems, but this cost is often neglected because it is less visible – for example, because it only indirectly affects customer satisfaction. System restarts and maintenance windows have become a generally accepted practice, even for business critical applications. For example, Microsoft Servers require restarting after certain updates of the operating system. But even payment systems are shut down several times per year 'for maintenance', which typically takes place between 00:15 and 02:15 at night in Belgium. On the site of Atos Worldline this is called a 'system stop', and *during these interventions, the payment network is not available and it is not possible to carry out electronic payments* [5]. This can be very inconvenient for the user if the restaurant bill has to be payed while the cards do not work, and neither does the ATM (automatic teller machine). Customer service could be improved if dynamic reconfiguration was made a design requirement for the payment system.

A. Maintainability improvements

Efforts to improve the flexibility and maintainability of automation systems go back decades. The first approach to implement automation control logic was based on hard-wired relay systems. In the late 1960s, GM Hydramatic issued a request for proposal for an electronic replacement. The result was a Programmable Logic Controller (PLC), built by Bedford Associates. One of the main advantages was that changes in control logic could now be made by changing the program rather than changing wiring and bypassing or adding relays. In addition, programs could be reused for another application [6]. The technology shift from hardware to software provided more flexibility and an improvement of maintainability.

Around the same time, dynamic reconfiguration was introduced as a feature of the *Multics* operating system. Multics is an acronym for *Mult*iplexed *I*nformation and *C*omputing *S*ervice. It was a mainframe operating system for which the design and planning started in 1964 [7]. It was commercialized by Honeywell and used till 2000. The goal was to ensure continuous availability of the mainframe running the Multics computing service, even when maintenance of physical components was required [8]. It was possible to switch between different hardware configurations, allowing for the replacement of CPUs (central processing units) and memory modules without switching off the system – resulting in dynamic reconfiguration.

While these certainly were improvements, the characteristic of true evolvability was (and is still) not totally reached. For example, many serious maintenance operations in the Multics operation system did still require a complete restart of the machine [9], and while changing or debugging a few lines of software code is easier than rewiring relay circuits, the number of software problems and bugs does not grow proportionally with the software size. Instead, they grow out of proportion. After reaching a certain size, software becomes a problem in its own right [10].

B. Standardized programming languages

For the reusability and portability of features from one (sub)system to another, a common programming language which is shared between these systems is an improvement over proprietary approaches. If possible, the use of standardized programming languages is most appropriate. The IEC 61131-3 standard [11] introduced standardized programming languages for PLCs. However, development environments which include compilers for these programming languages are typically proprietary systems and contain vendor-dependent differences.

In the world of information systems development, the impact of standardization is limited. There are thousands of programming languages. According to [12] more than 115 languages were implemented by 1968, while there were already over 1000 that had been used somewhere by 1999. Only a limited number of these have become relevant, but none can be considered as a global de-facto standard. This is confirmed by the different indices that measure programming language popularity, such as Tiobe [13]. According to Tiobe, the most used languages in August 2013 are Java and C, each having a market share of close to 16%, followed by C++ with a 9.4% market share. Several of these programming languages are defined in an associated (international) standard. As an example, there is ISO/IEC 14882:2011 on the programming language C++ [14]. For the Java language, no international standard is available, even if it is currently the most popular programming language. It also should be noted that only few compilers implement an entire language standard and nothing but this standard. An overview of the support for the C++ standard in different popular compilers is given in [15]. A similar situation exists in the world of relational databases. While an SQL standard is defined in ISO/IEC 9075-1:2008, multiple incompatible SQL dialects exist.

These implementation differences are a first obstacle to evolvability. In most cases it is not possible to simply replace a compiler with a new version from another vendor. Unless special middleware or libraries are used for decoupling (for example, Hibernate), database management systems cannot easily be replaced. Sometimes even the upgrade from an old version to the new version from the same vendor may cause run time errors. Similar issues arise when, for example, a web application framework like Apache Struts needs to be replaced by another framework.

C. Standardized approaches to organizing programs and data

The way how data and functionality are organized within the constructs provided by the chosen development environment is a design decision which is often left to the individual developer. However, in order to be able to share these constructs with other developers, common models can be useful: Not only programming languages, but also the way how data and programs are structured can be standardized.

At the design level, a number of standards are available. Their acceptance amongst practitioners is rather low, however. One of the few standards that stand out in terms of use is the Object Management Group's Unified Modeling Language (UML, ISO/IEC 19505). UML is a graphical modeling language for software engineering. The UML specification contains 14 types of diagrams, seven of which are (static) structure diagrams and seven are used to describe dynamic behavior.

One of the common modelling techniques for functionality is the finite state machine (FSM). FSM follow a straightforward syntax of states and transitions. Their formal semantics is well defined and based on a simple though rigorous mathematical model [16]. Generic in nature, FSM can also be combined with application domain knowledge for the purpose of standardisation. For example, the ISA-88 standard [17] recommends specifying elementary operations in batch manufacturing processes (e.g., filling a tank) by way of state machines. In the simplest case, the state machine for a so-called "equipment phase" contains the states "Idle", "Running" and "Complete". An equipment phase specification will, among other things, describe additional states, the conditions for transitions between these states (e.g., after a specified amount of time has elapsed) and the actions to take upon such a transition (e.g., close a valve).

While state machines are a valuable tool to increase system maintainability, they do not automatically guarantee evolvability. The research presented in this paper focuses on the design of an evolvable state machine. Such a state machine could be implemented in one or more of the IEC 61131-3 languages, independent of vendor or CPU type. It should be possible to base applications in information systems on this design as well. The design supports dynamic reconfiguration wherever possible.

The remainder of the paper is structured as follows: Section II gives an introduction on finite state machines, the underlying mathematical model and UML state diagrams. Section III explains the basics of Normalized Systems Theory (NST), which offers a formal guideline to system evolvability. This section also introduces an application concept, derived from Normalized Systems theorems, to support the co-existence of different versions of a program element in an evolvable system. Section IV presents three examples of state machines going through subsequent evolution steps. Section V develops an inventory of anticipated changes to state machines, illustrated by these examples. Section VI proposes a set of design rules for evolvable state machines in the context of the concept presented. Section VII discusses implementation considerations, giving special attention to hierarchical state machines, and Section VIII concludes the paper.

II. State machines

The concept of finite state machines (FSM) has its origins in work by McCulloch and Pitts in 1943, as part of their research on human cognition [18]. It was mainly through the work of George H. Mealy and Edward F. Moore, published respectively in 1955 and 1956, that FSM became popular as a design tool for digital systems.

An FSM is an abstract machine that can be used to model and/or specify sequential logic. A FSM can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called its current state. The current state can change upon a triggering event or condition. This is called a transition. A particular state machine is defined by the list of its states, the possible transitions between them, and the triggering condition for each transition.

This definition can be cleanly expressed by means of a simple mathematical model. Following [16], a deterministic finite automaton is represented by the 5-tuple

$$(Q, \Sigma, \delta, q_0, F)$$

with

- $Q = \{q_0, q_1, ...qn\}$, a finite non-empty set denoting the states the system can be in;
- Σ, the input *alphabet,* a set of symbols denoting the possible inputs;
- $\delta \in Q \times \Sigma \to Q$, the state transition function;
- q_0, the initial state;
- F, a set of final states.

The input alphabet Σ is the finite set of symbols that are accepted and can cause a transition. An input symbol represents an external condition. The state transition function δ determines the next state, based on the current state and input symbol. This is a partial function: The function is not defined for all possible combinations of states and input symbols. It is assumed that the next state will always be reached.

As far as system output is concerned, this model is quite limited: It can only describe an “accept/reject” result. If, for a particular sequence of input symbols, the automaton reaches a final state, the sequence is accepted; otherwise, it is rejected. This is not very useful for modelling the output of PLCs or most other software systems. For these purposes, it is helpful to extend the model so that it can describe the output that the system generates as it transitions between the states. The term ‘finite state machine’ is typically used to refer to an automaton with such output capability.

Traditionally, two different designs of state machines are distinguished: Moore machines and Mealy machines. In a Moore machine, the output is a function of the current state only, while in a Mealy machine, the output is a function of both the current state and current input. Moore and Mealy machines are special cases of the general model shown above. A Moore machine is represented by a 7-tuple

$$(Q, \Sigma, \delta, q_0, F, \Gamma, \omega)$$

with, just as before,

- Q, a finite non-empty set denoting the states;
- Σ, the set of possible inputs;
- $\delta \in Q \times \Sigma \to Q$, the state transition function;
- q_0, the initial state;
- F, a set of final states;

and, in addition,

- Γ, a finite non-empty set of output symbols;
- $\omega \in Q \to \Gamma$, the output, depending on the current state.

The representation of a Mealy machine is the same 7-tuple

$$(Q, \Sigma, \delta, q_0, F, \Gamma, \omega)$$

as for the Moore machine, but with

- $\omega \in Q \times \Sigma \to \Gamma$, as the output depends on both the current state and the input.

In practice, FSM are defined either in diagram notation (typically circles and arrows), or by a state transition table. In Figure 1, a simple Moore and a simple Mealy machine are shown as a UML (Unified Modelling Language) state

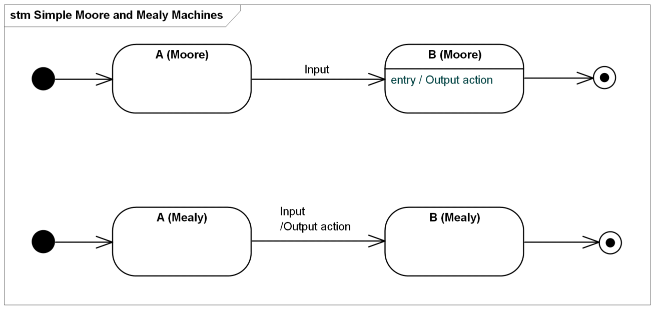


Figure 1. Moore and Mealy machine example

diagram. In UML, states are denoted by round-cornered rectangles, contrary to the classical graphical notations where circles are used. The black circle is the *initial state*, where 'the machine starts'. The circle with the dot is the *final state*. Transitions are denoted by arrows between two states.

For the Moore machine, outputs occur when the machine enters the new state. The transitions fire when the proper input is present. Therefore, the output action is shown on the state itself, and the transition is labelled with the input only. For a Mealy machine, every transition generates an output. Hence, every arc is labelled with both the input and the resulting output.

In a Moore machine, if different transitions take the finite-state machine to the same state, the same action is performed for all these transitions (since output is associated with the state). In a Mealy machine, if multiple transitions that have the same destination state should cause the same action, this action must be triggered by every transition on its own. Still, every Mealy machine can be converted to a Moore machine and vice versa, i.e., their mathematical models are functionally equivalent.

A. Implementation

Finite state machines grew popular in the world of digital circuit design. Both the next state and the outputs are calculated by means of a collection of logical gates (combinatorial logic). Figure 2 shows the respective designs for a Moore and a Mealy machine.

In a Moore machine implementation, the output must be calculated when entering the state. In a Mealy machine, the states are just 'resting points'. No more calculations are needed when the new state is entered. All actions occur during the state transition. In the mathematical models, it is not relevant when the output is calculated, since it is assumed that this calculation is instantaneous. However, the resulting hardware design will be different for a Moore machine and its Mealy equivalent. Typically Mealy machines have fewer states and are faster than Moore machines. Moore machines on the other hand are easier to program and require less output logic. Moore machines are often used for the design of controllers in digital circuits, while most software implementations use the Mealy model or a combined/extended model.

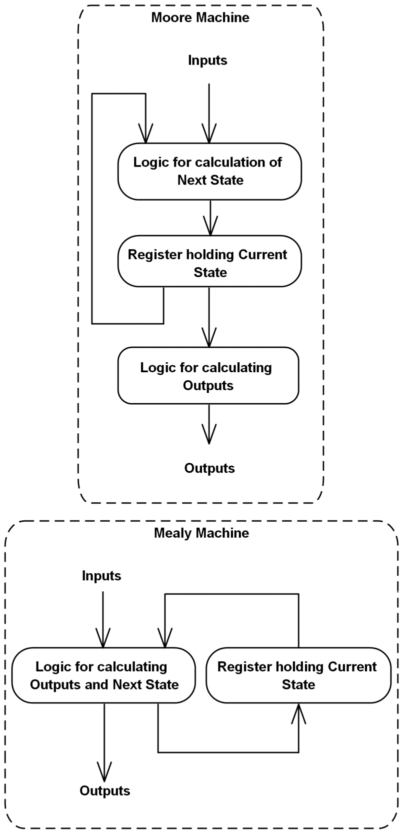

Moore Machine
Inputs
Logic for calculation of Next State
Register holding Current State
Logic for calculating Outputs
Outputs
Mealy Machine
Inputs
Logic for calculating Outputs and Next State
Register holding Current State
Outputs

Figure 2. Moore and Mealy machines

State machine implementations must consider how the 'real world' compares to the idealized model. For digital circuits, this is relatively straightforward: It is only required that a stable electrical output is obtained within a single clock cycle. If this is the case, in normal operating conditions, the implementation matches its underlying mathematical model. The properties derived from the model also apply to the digital circuit. Drawing the same conclusion for state machines in a software system in general will likely be more difficult. For example, the model assumes that no error occurs when calculating the next state in response to a trigger. To make the software system reflect the model accurately, either the state machine has to be made more complex to reflect these exceptional conditions, or a number of restrictions need to be imposed on the software system (for example, that calculations should not fail; or, in a Moore machine implementation, a new output should not be produced if the next state is not reached). Either way, a consistent software implementation of the simple mathematical models may turn out to be quite complex.

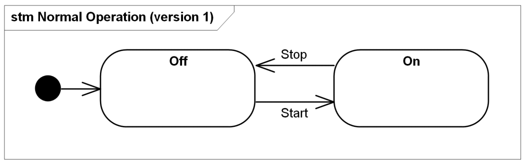


Figure 3. Motor control state machine, version 1

B. State space size and complexity

In automated production installations, the behaviour of a large portion of the control equipment (such as valves, light curtains, pumps, mixers or conveyers) can be – and is – modelled using state machines. Figure 3 shows a very simple model of motor control system, which could be implemented on a PLC. The motor has two states, 'On' and 'Off' and an initial state. Two conditions affect the state of the motor: the 'Start' and the 'Stop' command. The output (whether the motor runs or not) is fully determined by the state, hence this is a Moore machine.

Typically, a factory contains more than one motor. In case this state machine is extended to control multiple motors with a single controller, its 'state space' increases. The 'state space' is defined as the collection of all possible states the controller or system can be in. The state space of Figure 3 contains two states. In case this state machine is extended to control two motors, there are four states (On—On, On—Off, Off—On, Off—Off); a third motor would yield a state space of eight states. An installation with 100 items to be controlled, each for example having 10 states, yields 10x10x…x10=10^{100} different states, an installation with 1000 items 10^{1000} states. The fact that the number of states grows exponentially with the size of the installation is called *state space explosion*. It clearly is not practical to model a large system this way.

State space explosion is often used as an argument against the use of state machines by software designers who are not familiar with the concept. However, nobody would ever design a 'flat' controller with 10^{1000} states. The solution to this problem is twofold. First, the overall design can be split into a collection of communicating state machines; in our example, a separate instance of the original state machine for each motor. Second, states can be grouped hierarchically.

However, careful design of the communication between the state machines is required: Wagner and Wolstenholme [19] point out that *sending unconstrained messages to another state machine is like a jump to another program part* using a 'goto'. As state machines run concurrently, issues such as deadlocks may arise. In order to create maintainable state machines, the messaging between the state machines needs to be restricted.

C. UML statecharts

In 1987, Harel [20] introduced 'statecharts' as an extension to classical state machines, supporting hierarchy and concurrency in a single graphical formalism. Harel's statecharts form the basis for state machine diagrams in the UML standard, which includes them as a means to specify the behaviour of a software based system. Statecharts introduced hierarchical or *composite* states to enhance the readability of complex state diagrams. A system is regarded as an abstract state machine with a limited number of states. Every state can be further refined by defining another state machine within that state, supporting a top-down design approach. The UML standard does not specify up to which level the states must be refined. Thus, in a UML statechart, 'not all states are equal'. Some states can be modelled down to the level of boolean logic, other states may be a black box implemented by an entire business application.

In UML statecharts, transitions are caused by 'triggers'. The trigger corresponds to the input symbol in the FSM model, but can be any signal, event or change in a condition. The transition can be conditional to a 'guard'. The result of the transition can be twofold. First, the transition can have an 'effect', an action that is executed conditional to the firing of the transition. Second, the transition results in a state transition which can also cause a number of actions to be performed. An *exit* action can occur when the current state is left; an *entry* action may be executed when entering the new state. In UML, the order in which these actions are performed is well defined (exit action first, then transition effect, then entry action). Likewise, when hierarchical states are used, the order in which all actions need to be performed is well defined. A basic assumption is that a state machine can only start processing an event if it has finished processing the previous event.

Apart from the hierarchical definition of states, UML statecharts have a much richer syntax than classical state machines have, such as constructs to model concurrency through forks, joins and regions. Through the use of 'pseudo states' such as a junction, an entry and exit state and even a 'history state', the communication between the different state machines in a design can be specified.

Because of this added complexity, a formal definition of the semantics of UML state diagrams is much less straightforward than the simple mathematical model defining the semantics of finite state machines. This almost necessarily results in a number of ambiguities in the specification. Regarding the latter, [21] reports that 29 new unclarities were introduced in the UML 2.0 state machine specifications. In [22], an axiomatic semantics of these UML 2.0 state machines is provided by giving solutions outside the UML standard. According to [23], the current UML standard 2.4 introduced 6 aditional ambiguities.

In this paper, we study design rules for evolvable state machines. Even though we make use of UML, it is not our intention to analyse and discuss all the syntactical features available in UML statecharts. Our use of UML is limited to the capabilities of a classic FSM extended with simple composite states. UML statechart constructs dealing with concurrency and complex communication will be the subject of future research.

III. Normalized Systems Theory

Software undergoes an ageing process, as recognized by Parnas [24]. Since there are indications that this ageing process is also happening with business processes [25], we must consider the possibility that this phenomenon may actually apply to all non-physical systems in general, which undergo an evolution in our society and economy.

Earlier work pertaining this subject matter was done by M. Lehman, resulting in his 'Laws of software evolution'. He formulated the law of increasing complexity, expressing the degradation of a system's structure over time [26]:

"As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it."

Based on his work for IBM on the design and implementation of the OS of IBM mainframes in the late 1960s, Frederic Brooks [27] made the observation that

"Program maintenance is an entropy-increasing process, and even its most skilful execution only delays the subsidence of the system into unfixable obsolescence."

Ever since, a myriad of software engineering methodologies were invented, new programming languages were created, paradigms developed from 'structured' over 'object oriented' to 'aspect oriented' programming. This did not, however, fundamentally change the issues related to software evolvability. Brooks stated that because of the very nature of software *no inventions will do for software productivity, reliability, and simplicity what electronics, transistors and large-scale integration did for computer hardware* [28]. Part of the complexity is caused by constant pressure for change, imposed by continuous change of the environment in which the software system is embedded.

In software development, every change causes a further deterioration that progresses with each update or hotfix. Over time, the deficiency of the structure renders the system unworkable. To mitigate this problem, a re-write of the whole system can help (Figure 4 [29]).

The theory of Normalized Systems was introduced to challenge Lehman's law [30]. Other than most previous efforts to achieve maintainability of software, the contribution of the Normalized Systems Theory goes beyond heuristics; instead of only advocating guidelines such as "low coupling and high cohesion" (these are are widely accepted heuristics, e.g., [31]), it provides theorems to derive yes/no answers to questions about evolvability.

In the context of Normalized Systems, an action entity shall be defined as a module which contains functionality,

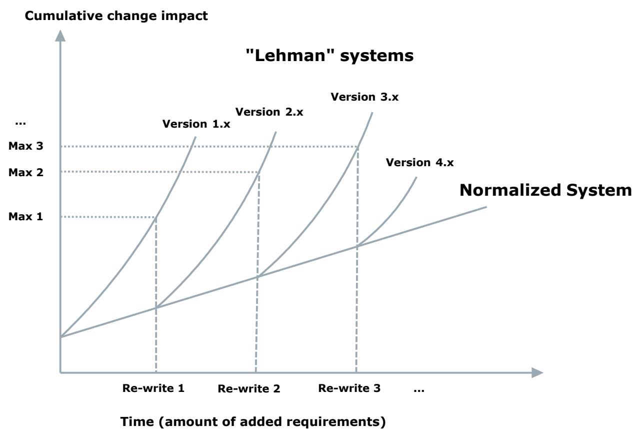


Figure 4. Improving software structure with a re-write [29]

and a data entity shall be defined as a set of tags (fields). Action entities and data entities are the two main elements from which a system can be constructed. Action entities use data entities as input and output parameters. States, conditions, commands or events can be stored in a data entity. The four core theorems of Normalized Systems are:

1) Separation of concerns: *An action entity can only contain a single task.*
   A task is functionality which can evolve independently. If the system's developer anticipates that two or more parts of the core functionality can change independently, these parts must be separated. Therefore, Normalized Systems shall be constructed of action entities dedicated to one core activity.
2) Data version transparency: *Data entities that are received as input or produced as output by action entities must exhibit version transparency.*

   It must be possible to update one or more data entities which are passed between action entities and let multiple versions co-exist without affecting other versions of action entities.
3) Action version transparency: *Action entities that are called by other action entities must exhibit version transparency.*

   It must be possible to update an action entity, which is coupled with another action entity, while multiple versions of both modules can co-exist. In other words, introducing a new version of an action entity shall not require changes to any other action entity.
4) Separation of states: *The calling of an action entity by another action entity must exhibit state keeping.*

   Every action entity must keep track of its requests to other action entities. If the response to a request is not as expected, the calling action entity must not block indefinitely; rather, it shall handle the exceptional situation as appropriate for its own state.

Two additional theorems have recently been introduced as extensions of the theorems on data and action version transparency [32]. They address the challenge of managing the diversity of run-time *instances* of data and action entities in an evolving system.

5) Data instance transparency: *A data instance has to keep its own instance ID and the version ID on which it is based or constructed.*

   If the type definition (source code) of a data entity is updated to a new version, instances based on the previous version continue to exist in the system. If an action entity receives a data instance for processing, the action entity must have a way of knowing the version of this data instance to be able to handle the data instance in a version-compliant way. Therefore, every data instance must contain a version ID reflecting the version of the data entity it is an instance of. The instance ID serves to tell apart multiple instances of the same version.

6) Action instance transparency: *An action instance has to keep its own instance ID and the version ID on which it is based or constructed.*

   When run-time instances of action entities interact, they must consider the fact that the other action entity can be based on one of various versions of its type definition. A version ID is necessary to give the calling action instance information about which interactions are possible. Again, the instance ID serves to tell apart multiple instances of the same version.

A. Applying instance version diversity

In order to create an evolvable software system, the above-mentioned rules must be respected throughout the entire design and implementation. In the following, we introduce an architecture supporting version diversity in PLC systems as an application case study. It is designed to support the implementation of cross-vendor PLC systems, allowing for the co-existence of multiple versions of all hardware and software components, and to support true dynamic reconfiguration. Version diversity is supported with respect to:

- Application functionality
- Vendor dependencies in PLC programming
- Vendor dependencies regarding the control of field devices (e.g., a motor).

When an update of a PLC project includes the introduction of a new CPU type, or even a conversion of the software to another brand, software developers often re-engineer the whole project. Also, when a motor is replaced with a different one – or the update includes the introduction of a frequency drive, which requires a vendor dependent system function block –, engineers tend to re-write the module which is controlling the motor.

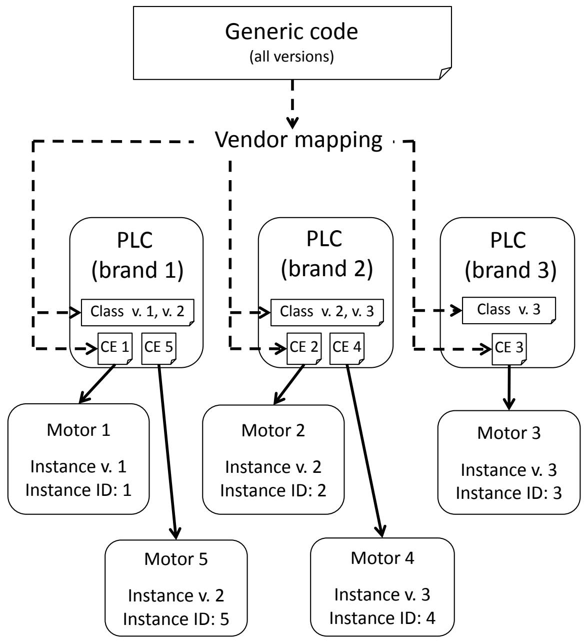


Figure 5. A generic software module with heterogeneous instances

The architecture presented intends to reduce the amount of these re-writes. A new motor should no longer require a new software module; neither should the entire project require re-engineering because of changing to a new brand or PLC family. To achieve this, we propose that programming is based on generic modules. There shall only be one module for every core function (e.g., motor control), with the variations between physical motors and their control being addressed by multiple, co-existing, versions of this module.

In this concept, the vendor independence brought about by the IEC 61131-3 standard is an important element. Nowadays, most common brands support at least some of the IEC 61131-3 languages. However, the standard does not include hardware configuration. Consequently, the connection to process hardware (process I/O) remains vendor-dependent. In addition, the standard allows some liberties (e.g., implementation-dependent parameters in Annex D [11]). Commercial IEC 61131-3 programming environments show some differences. Therefore, developers still often re-write a whole software project in case another brand of PLC is required.

Our approach to truly generic, vendor-independent PLC programming is shown by way of an example in Figure 5. A generic module, which strictly sticks to IEC 61131-3 code, contains the core functionality of a device, for example controlling a motor. Instances of this module represent individual motors. Before the generic module can be downloaded to a specific brand of PLC in order to control

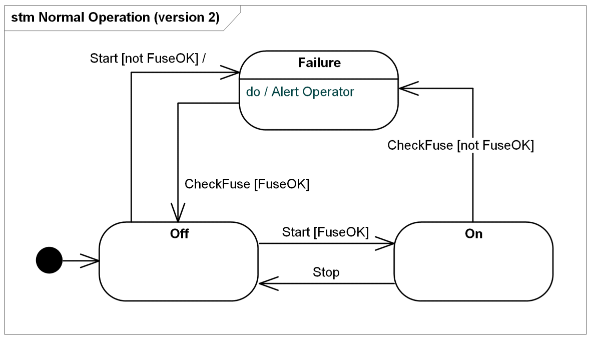


Figure 6. Motor control state machine, version 2

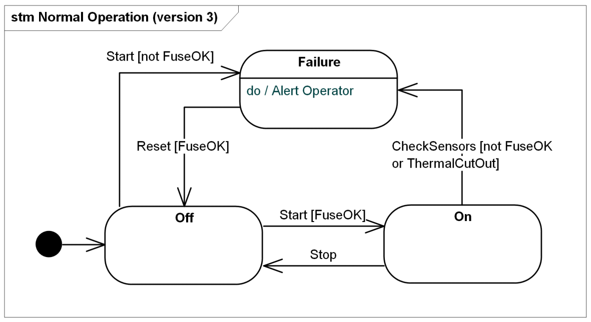


Figure 7. Motor control state machine, version 3

a specific motor, it undergoes an automatic vendor-mapping procedure, which converts part of the module code according to what is required by the vendor's specific environment. In addition, the vendor-mapping procedure adds an extra module to the (mapped) core module: a connection entity (CE). This connection entity is dedicated for a specific motor (instance), and includes all the details needed to connect the (mapped) core functionality with the process hardware (I/O). If necessary, this connection entity can also include vendor-specific function blocks (e.g., a scaling block for analog values, or a system block dedicated to control a specific frequency drive).

Each new version of the functionality is referred to as a class version, and each individual physical motor as an instance. Class versions correspond to the functionality *available* in the PLC (potentially in several co-existing versions), while instance versions correspond to the *instantiated* functionality for controlling a specific type of motor. Instance IDs refer to one single, specific physical motor; they tell the connection entity which hardware addresses an individual motor control module instance has to be connected to.

IV. State machines and evolution

In this section, we discuss examples for how state machines could evolve in response to new requirements and hardware capabilities, leading to new versions of the state machine.

A. Motor control

The state machine in Figure 3 is a very idealized and simplistic view. In an actual industrial environment, additional conditions such as failure conditions or interlocks must be taken into account. As an example for such an additional condition, the second version of the state machine considers the condition of a fuse.

In version 2, the start condition becomes 'Start and FuseOK' (Figure 6). In UML syntax, we model this by extending the 'Start' trigger with the 'FuseOK' guard, making the transition conditional on the state of the fuse. The additional state 'Failure' is introduced. The condition for transitioning from the 'Off' state to the 'Failure' state is 'Start and not FuseOK', i.e., the transition is triggered by the 'Start' event, also conditional to the state of the fuse. If the fuse blows during the 'On' state, a transition to the 'Failure' state results. To detect the state of the fuse, a 'CheckFuse' function is needed, being the trigger for the transition from the 'On' state to the 'Failure' state. In addition, when entering the failure state, a notification is made to trigger an operator to solve the issue, represented by the action 'Alert Operator'. The condition to go from the 'Failure' state to the 'Off' state is 'FuseOK', the trigger is 'CheckFuse'. This implies that the engine will restart automatically when the fuse is repaired.

The third version considers the situation that the motor can stop due to a thermal cut out in the 'On' state (Figure 7). This is modelled with a new version of the transition to the 'Failure' state. In addition, the operator must push a reset button before the 'Off' state can be entered again after a failure. Hence, an additional 'Reset' trigger is introduced resulting in a new version of this transition.

In version 2, only the state of the fuse needed to be checked, by means of a function 'CheckFuse' triggering the transitions from the 'On' state to the 'Failure' state and from the 'Failure' state to the 'Off' state. In the third version, two conditions need to be checked: the state of the fuse and a heat sensor connected to the motor. Depending on the implementation, this may require the creation of a new version of the trigger that can deal with multiple sensors.

B. ATM example

This example introduces the use of hierarchical states in UML state machine diagrams. It shows a high level specification of an ATM from a user perspective, limited to the withdrawal of money (Figure 8). As in the motor control example, two alternative versions of the initial design will be discussed, each resulting from changing requirements.

The process modelled in this example starts when a user inserts a bank card. This user must first authenticate. If the correct PIN (personal identification number) code is supplied, money is dispensed (Figure 8). The state machine

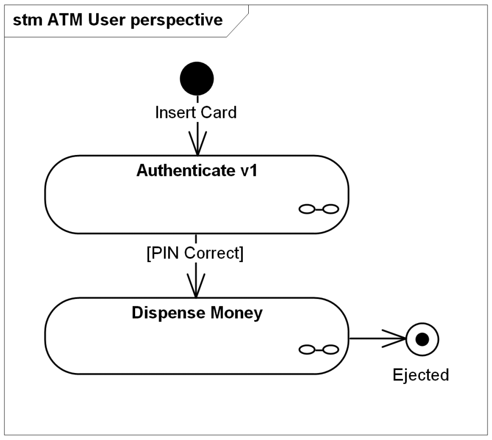


Figure 8. ATM specification using composite states

has two states: 'Authenticate' and 'Dispense Money'. The transition between the states is triggered by 'CheckPIN', a function which is started for example after having pressed the 'OK' button on the ATM. In this very simple example, the user is not given the option to select the amount of money withdrawn.

There is a major difference with the previous state machine examples. Both states in these examples are not simple, fully defined states as in the motor control example. They are in fact state machines themselves. Both states are therefore modelled as composite states, indicated by the 'infinity symbol' in the example. In Figure 9, detail for the 'Authenticate' state is provided.

The authentication starts with the processing of the card. When the machine is in the 'Read Card' state, the card is processed by reading and decoding the chip or magnetic strip on the card. Once the card is processed, the Authenticate state machine will automatically transition to the 'Enter PIN' state. Note that no trigger or guard is specified on this transaction.

When the machine arrives in the 'Enter PIN' state, the entry action of the state is executed. As a result, the machine will ask the user to enter a PIN code. When the user presses an 'OK' button on the ATM, 'PIN Entered' is triggered and the PIN code will be validated using the 'check PIN' action. If the PIN code is not correct, the user must try again, else the final state of the 'Authenticate' super state will be reached. When this final state is reached, the 'Authenticate' state itself is considered as complete and the system can transition to the 'Dispense Money' state. Once the money is dispensed, the card will be ejected and the final state of the overall process is reached.

The diamond within the 'Authenticate' state machine is a 'choice pseudostate'. This state machine could also be drawn with two seperate transitions with the same trigger as in the motor control examples above.

In this version, the ATM has serious drawbacks. First,

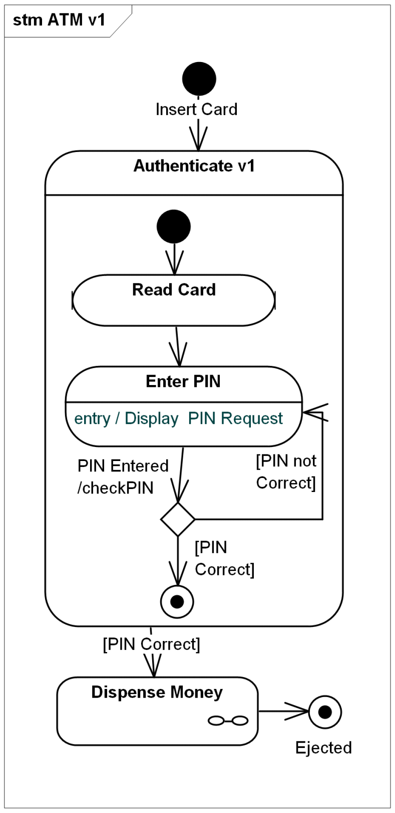


Figure 9. ATM version 1

the user can supply an unlimited number of PIN codes. Second, the card will only be returned after having supplied the correct code. To amend the first problem, the concept of an 'error counter' is added. Two actions to increase and reset this counter are added. If the card is unreadable or if the maximum number or retries is reached, the card is ejected. The user is also given the opportunity to cancel the session and eject the card (Figure 10). In these three cases, the 'Authenticate' state machine transitions to the state 'Transaction cancelled', modelled as an exit state. At the higher level in the state machine hierarchy, this exit state unconditionally transitions to the final state 'Ejected'.

In both versions the internals of the states 'Read Card' and 'Enter PIN' are not specified. To implement the ATM system, more detailed specifications are required, possibly resulting in additional levels of state machines and sub-states. Changing requirements may cause an evolution of the internals of these states, resulting in a third version of the ATM example.

As mentioned above, in version 2, 'PIN Entered' was triggered by pressing an 'OK' button on the ATM. Assume that in a third version of the ATM, the use of this button should be eliminated if the number of digits of the PIN codes can be derived from the card type, e.g., 4 digits for a Belgian bank card. The 'Read Card' implementation should

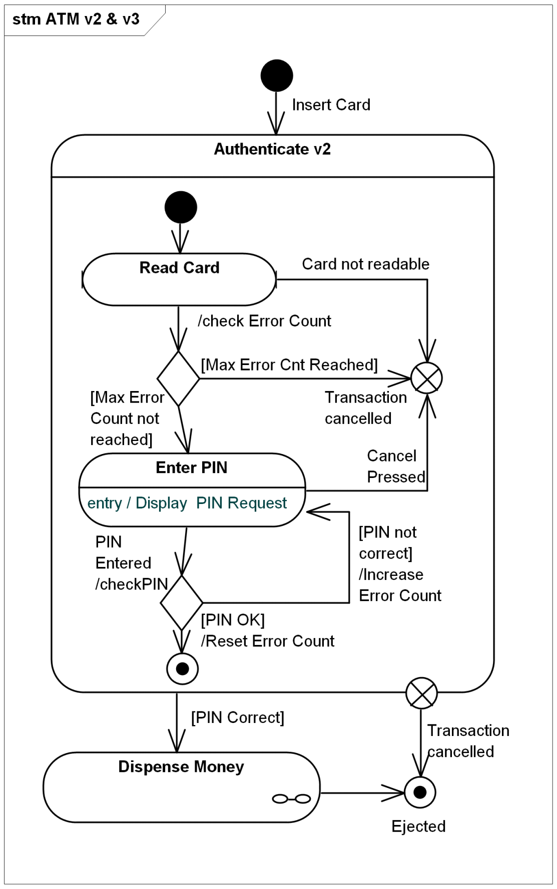


Figure 10. ATM versions 2 and 3

be enhanced to determine the type of card. The 'Enter PIN' implementation should be changed such that

- the entry action becomes conditional on the card type ('Enter PIN' or 'Enter PIN and press OK');
- for a Belgian card the 'PIN Entered' trigger fires once the fourth character is entered, and
- for any other card, pressing the 'OK' button causes this trigger to fire.

In this new version, the 'Authenticate' state machine itself does not evolve to a new version, only the two sub-states 'Read Card' and 'Enter PIN' are changed. Also, a new parameter needs to be passed, since the card type is detected by 'Read Card' and must be used by 'Enter PIN'. However, these differences do not lead to a change in the diagram. Therefore, Figure 10 does not change for version 3 of the ATM.

As in the motor control example, it is possible that both versions of the sub states need to co-exist for a certain time. Assume that the 'Enter PIN' process requires a software update, while the adaptation to 'Read Card' requires the physical installation of new hardware. Ideally, the software is upgraded on all ATMs, irrespective of the version of the card reader.

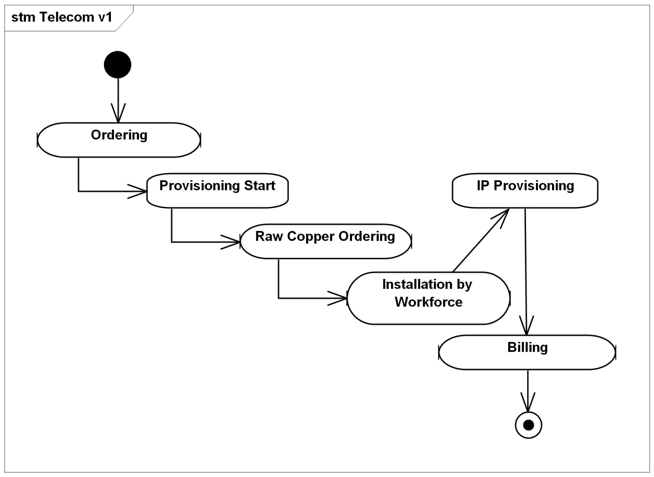


Figure 11. Telecom process, version 1

C. Integration of COTS applications

In a third example, additional requirements on the evolvability of state machines will be illustrated. This example is based on the experience of one of the authors with a greenfield software implementation and integration for a Dutch telecom operator. Some of the outcomes of this project for the academic community are documented in [33] and [34].

Running a telecom operation requires the automation of numerous very complex business processes. Most operators resort to the installation of Commercial Off-The-Shelf (COTS) applications instead of implementing everything in-house 'from scratch'. Often multiple COTS applications are installed, such as a Customer Relation Management (CRM) system, a telecom billing system, fraud detection, a network inventory system, an ERP system, etc. These COTS applications are large software systems, the installation of which requires extensive parametrization and configuration to implement the desired business processes. Typically, a multi-million dollar budget is required for the installation and implementation of a single COTS application.

To create end-to-end workflows supporting the different business processes, the different COTS applications must be integrated, this being a bespoke development for the operator. The integrated workflows can be modelled using hierarchical state machines. The highest levels of the hierarchy are part of the bespoke development by the operator. At a certain level in the hierarchy, a substate will be fully contained in one of the COTS applications.

Figure 11 shows a simplified order-to-bill process for a new DSL line, stripped from all triggers, guards and actions. In this example, an order for a new DSL line is treated first in the CRM system. Once the order is completed, it needs to become available in a 'service provisioning system'. The

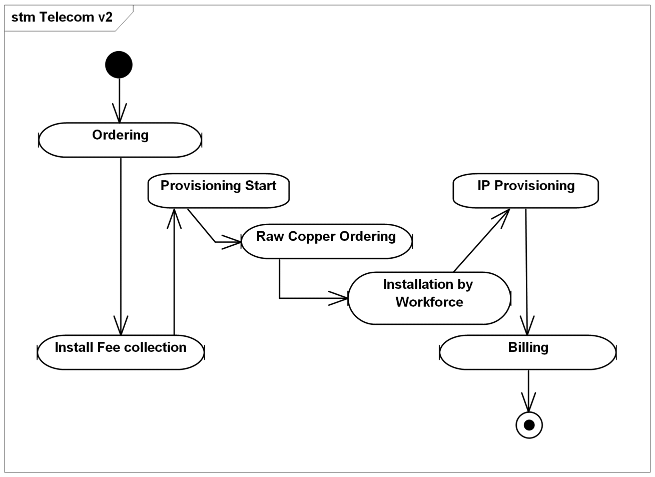


Figure 12. Telecom process, version 2

latter is used to manage the installation of the new line and ensure that the appropriate services are provisioned. The provisioning system needs to communicate with a system of another operator to order raw copper. After delivery of this line, the DSL line is physically installed at the customer premises, a process that is managed by means of a ‘workforce management system’. Next, the IP service is provisioned using the service provisioning system. Finally, once, the system must ensure that the appropriate billing information is created in the telecom billing system.

In Figure 12, a second version of this process is shown. In this adapted process, the customer must pay the installation fee before the provisioning starts. Once the service is provisioned, the recurring part of the billing must be activated. When the state machine arrives in the ‘Install Fee collection’ state, it will hand over the control to the billing system where an invoice will be created and sent to the customer. Next, the entire process will halt, waiting for the customer to pay. Only when the appropriate amount is collected, control will be handed over again to the higher level state machine.

The evolution of version 1 to version 2 is a complex process:

- The entire software installation runs on a large number of servers; the implementation of updates to any of the COTS systems is complex and time consuming. It is nearly impossible to shut down and restart everything at the same time without affecting user experience and/or operations. Attaining the ability of ‘dynamic reconfiguration’ would be a significant advantage.
- Every substate in the process requires a considerable period of time to complete. During the execution of the overall process for any given order, one or more updates and new versions may be introduced. Depending on a number of criteria, existing orders may still need to be handled according to the the old version. For example, the installation fee can only be collected before the installation if this was required during the ordering phase; if not, the order will need to be processed using version 1 of the state machine.

V. Changes to state machines

Industry usually prefers the Mealy state machine model over the Moore model, possibly because a Mealy machine often ends up with a lower amount of states, and is thus more compact. In contrast, Normalized Systems Theory advocates a higher granularity of modules. If ’more compact’ results in a lower amount of (larger) modules in case a Mealy machine is implemented, this would favour the Moore type.

Also, a Mealy machine combines transitions with actions, which can result in violations of the separation of concerns theorem: When a new version of a transition is implemented in the same module as the corresponding, resulting action, both original and new version need to co-exist. When a new version of the action is introduced afterwards, this update has an impact on both versions of the co-existing modules containing both actions and transitions. Moore machines explicitly separate states and transitions. For the implementer, it is easier to comply with the separation of concerns theorem if states and transitions are separated in the model the software is based on.

The preference of industry for Mealy machines is understandable: a lower amount of modules probably results in a lower implementation effort due to the lower amount of module definitions and module interfaces. However, changes to a single transition or to a single action might result in combinatorial effects when transitions and actions are implemented in the same module.

In order to expand on these general observations regarding the evolvability of state machines, this section is dedicated to a detailed examination of anticipated changes, illustrated by the examples in the previous section. As already indicated at the end of Section II, only a subset of the constructs available in a UML statechart is studied.

A. Identifying singular change drivers

The following changes can be observed for ‘flat’, single level state machines:

- adding a state (e.g., ‘Failure’ in Figure 6)
- adding a transition (e.g., ‘Card not readable’ in Figure 9)
- changing a state (e.g., change of ‘Enter PIN’ entry action from ATM version 2 to 3)
- changing a transition (e.g., the ‘On’ $\rightarrow$ ‘Failure’ transitions in Figures 6 and 7).

Neither adding/changing a state nor adding/changing a transition are singular actions, since they may involve a number of independent actions. Adding a transition may require the addition of a trigger, a guard and an action. Adding a state may require the addition of two transitions, an entry and an exit action. Changing a state may require additional data to be passed between states. According to NST, change drivers need to be singled out to study evolvability. Hence, the above-mentioned anticipated changes need to be further refined.

Likewise, neither changing the direction of a transition nor changing the target of a transition are independent actions. Both consist of adding a new transition together with simultaneously deleting the original transition. As will be discussed below, deletions are not desirable from an evolvability viewpoint, which is why they are not included in the list above.

For reasons of simplicity we will not further take the 'guard' of a transition into account. It is assumed that the trigger event will occur conditional to the value of the guard. We consider this as a further restriction on the subset of UML state machine constructs that are within the scope of our current analysis. A detailed analysis of this subject matter shall be part of future research.

While triggers and actions can be easily distinguished as singular change drivers, it is more difficult to define a 'singular addition of a transition'. In its most basic form, a transition $A \rightarrow B$ unconditionally fires when the state A becomes current, and causes no effect other than putting the state machine in this next state B. On arrival in state B, actions related to this state will of course be executed.

The addition of an unconditional transition to an existing state machine may cause inconsistencies, however. Assume, for example, that the transition from 'Read Card' to 'Transaction cancelled' in Figure 10 were unconditional. The resulting machine would always transition to the final state 'Ejected', which is of course, undesirable. There are many solutions to this problem. A possible solution, requiring the introduction of concurrency in the design, is the following: Immediately after reading the card, execution should split in two parallel, concurrent paths. A first path goes to an end state by ejecting the card. In parallel, the process continues as if the card was read. While this change to the design of the state machine yields the desired result, it clearly is not a singular change.

To find a definition for the singular addition of a transition, we shall resort to the mathematical model underlying the state machines. In this model, the state transition function

$$\delta \in Q \times \Sigma \rightarrow Q$$

maps the combination of a current state and an input to a next state. Not every combination (current state, input) is defined. Only a limited number of inputs are accepted in any given state. Every combination (current state, input) can only occur once, i.e., the next state is defined unambiguously.

In this model, adding a transition cannot introduce an inconsistency in the state machine as long as the restriction is met that the next state is defined unambiguously. We therefore impose additional restrictions on the UML state machines we consider in order to replicate this characteristic, and limit ourselves to state machines without concurrency that only have conditional transitions. Hence, a singular change driver can be obtained when the transition has a trigger and under no circumstances a condition could arise where two transitions need to fire concurrently. Note that the same restriction holds for every addition and change of a trigger.

These are significant constraints on the usual semantics of UML statecharts. Nonetheless, in the remainder it is implicitly assumed that these consistency requirements are respected through every step in the evolution. Hence, they will not be part of the design rules we will propose.

This results in the following anticipated changes to a flat state machine:

1) adding a transition (maintaining the consistency requirements) between two states, without effect
2) adding a trigger to a transition (changing/deleting)
3) adding an action to a transition (changing/deleting)
4) adding an empty state, without internal actions
5) adding an entry action (changing/deleting)
6) adding an exit action (changing/deleting)
7) passing additional data between states.

In case of non-flat, hierarchical state machines, the internal behaviour of the states will become more complex. In the telecom example, the 'Billing' state, comprising a very complex state machine in a COTS application, was split in two parts. Part of the actions executed in the state 'Billing' will be separated amongst the new states. Part of the action, such as the creation of the invoice and the collection of the payment, will need to be supported in both states. A number of anticipated changes can be derived:

1) adding a state machine to a flat state
2) changing an existing state machine
3) deleting a state machine to return back to a flat state.

For the purpose of our discussion, we only consider a restricted set of hierarchical state machines which can be flattened. Therefore, composite states only have a notational purpose. Since changes to them can be expanded to changes on the equivalent flattened state machine, they are not further considered as anticipated changes. If certain restrictions are lifted, and e.g. concurrency is allowed, however, this will need to be reconsidered.

Apart from the entry and exit actions of a state, the UML standard also specifies 'internal transitions'. These are transitions that cause an output action but do not cause a state transition. These transitions fire as a response to a specific event if the machine is in a given state. This type of transitions has deliberately been excluded from the discussions above. We consider them as special cases of hierarchical state machines. From a semantical point of view, this is a further restriction we impose. It will, however, not impact the conclusions listed below.

B. Changes and version transparency

Deleting a state, transition or any other element usually is intuitively considered an anticipated change to a design. However, according to NST, a deletion violates the action version and data transparency theorems and should therefore never be allowed. Assume that in the ATM example, the bank card technology evolves in such a way that the number of digits to be entered can be read from the card using a new card reader. This of course requires an update of the card reader which is used in the 'Read Card' state. In a fourth version of the state machine, 'Enter PIN' would then make use of this number rather than the card type. If, however, the card type parameter is deleted, versions 3 and 4 cannot coexist any more. Similar reasoning holds for the deletion of transitions and states. Changing the direction of a transition is in essence (a) the addition of a new transition in the opposite direction and (b) the deletion of the original transition, and should therefore not be allowed either.

To keep systems compact nonetheless, NST proposes an extended 'garbage collection' approach on the design. The removal of states, transactions, or other constructs can make it impossible for older versions to exist. Consequently, deleting a construct should only be allowed if one can guarantee that no existing module makes use of it. In the ATM example, this means that the card readers have been physically upgraded for all ATM machines in the field. Note that the knowledge whether this has been done or not *cannot* be derived from the design itself.

Just like a 'delete', a 'change' can cause a violation of the version transparency theorems. Note that confusion can arise with restricting 'change', because the area of concern of NST is 'change' of an evolving system without causing combinatorial effects. In this paper the theory is applied to state machines. The goal of NST is to determine how a design and/or implementation can be *changed* in a way that a continuous and potentially infinite evolution of the system can be achieved. To avoid this confusion, we prefer to use the term 'modification' when we refer to an update of the functionality or data structure of a construct. To prevent violations of the version transparency theorems, modifications should be based on the concepts of transparent coding or version wrapping [35]:

1) Transparent coding: Transparent coding is defined as inserting internal code in a module which does not affect the functionality of previous versions. Since Normalized Systems require high granularity, it is not unexpected that the individual (small and straightforward) modules or subroutines end up to be a simple piece of code, on which the programmer has a clear overview. In such cases, the programmer can preview the effect of a functional change on previous versions, and maintain downward compatibility. If the change is not contradictory with one of the previous versions, it might be possible to apply transparent coding.

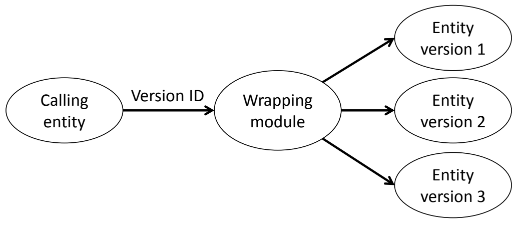


Figure 13. The concept of version wrapping

This means that the new functionality can just remain in the module without affecting the original code, even if a calling entity is not aware of the new functionality.

2) Version wrapping: There will be lots of cases where transparent coding is not possible, because the code is too complex for the programmer to have a reliable overview, or if the new functionality is contradictory with one or more of the previous versions. To exhibit action version transparency, the different versions can be wrapped. The calling action has to inform the called action which version should be used by way of a version tag. In addition, following the separation of states theorem, the called action has to inform the calling action whether the instance of the called action is recent enough to perform the requested action version.

An action entity which is designed according to the concept of version wrapping aggregates functionality for all the versions as separate action entities. Each of the nested action entities contains a version of the core functionality. Following the separation of concerns theorem, the wrapping action entity should not contain any core functionality, but limit itself to wrapping the versions as a kind of supporting task. Following this principle, different versions of a module co-exist in parallel. Figure 13 illustrates how a wrapping module selects the desired version based on the version ID.

C. Anticipated changes

In summary, in order to attain the property of evolvability for a state machine, updates must be confined to the following set of anticipated changes:

- An additional state
- An additional transition
- A new version of a state following the principle of transparent coding
- A new version of a transition following the principle of transparent coding
- A new version of a state following the principle of version wrapping
- A new version of a transition following the principle of version wrapping
- An additional entry/exit action of a state
- A new version of an entry/exit action of state
- A new effect (action) of a transition
- A new version of an effect (action) of a transition.

As mentioned above, adding a transition or adding a state may result in state machines behaving inconsistently. The introduction of a new transition or a new version from a state 'A' may impact all transitions having 'A' as current state, as the conditions on these transitions may need to be changed for consistency reasons. Hence, new versions of all these transitions will need to be created. Similarly, adding a new state 'N' requires the addition of a new transition from a state 'A' to 'N' and from 'N' to some other state, unless if 'N' is a final state. This implies that new versions may have to be created for all transitions that leave from state 'A'. In order to arrive at a new, consistent state machine, multiple of the abovementioned anticipated changes may be required. If however, the implementation of every single anticipated change respects the rules of NST, the overall change will as well.

VI. Derived rules for evolvable state machines

The previous sections discussed the benefits of version transparency and co-existence. In the following, we propose rules – based on Normalized Systems Theory – that shall be followed by state machines and the program code implementing them in order to achieve these properties, and, thus, evolvability.

S1. *The functionality of a state machine shall be implemented in an action entity, while the state and transition trigger information should be stored in a separate entity – a data entity.*

When the functionality of a state machine is updated, a new class version is introduced. We want to be able to deploy instances of this new version to the system without disrupting the operation of instances of previous versions, which may still be adequate for part of the equipment. Thus, several instances of devices controlled by different versions of the state machine should be able to co-exist, and we require the logic of the state machine to change independently of these instances. Remember that if two parts of a module can change independently, they shall be separated following the separation of concerns principle (Theorem 1); from this follows the separation called for in this rule.

Every version of the data entity contains state and condition fields. In each state, a particular state of the associated process hardware is effected (e.g., letting the motor run in the 'On' state). This is done by way of a connection entity. Values for the condition fields are provided by other action entities (in particular, connection entities); when the conditions for a particular transition are fulfilled, the state machine action entity changes the current state of the instance.

In UML, a trigger is an action responding to an event that occurs outside of the state machine. According to rule S1, the state machine only contains a generic mechanism to determine what transition to execute based on the contents of a data element.

S2. *The state machine data entity shall include an instance ID.*

The instance ID allows the connection entities to map this instance to the underlying system. In a PLC, this mapping would contain the hardware addresses; in a business process implementation, the mapping could involve the details of interfacing with a COTS system. The mapping is necessary so that changes in the underlying system (e.g., on hardware inputs) are reflected as changes in the transition fields, which in turn will cause the state machine action entity to perform the appropriate state transition. Likewise, the mapping is necessary in order to command the underlying system to perform the action associated with a state of the state machine (e.g., setting the required hardware outputs).

S3. *The state machine action entity shall include a class version ID, and the state machine data entity shall include an instance (data type) version ID.*

To comply with the version transparency theorems, the data entity must contain its own version (the version of the state machine it is an instance of). This version ID lets action entities recognize the class version corresponding to the instance and act accordingly. The action entity should store its class version on the moment of compilation as a hard-coded constant.

Following our first rule (S1), the data and the functionality within the system should be separated. Therefore, we have a data entity to store the system's data in one or more data fields, and an action entity to perform actions based on the data in this data entity. Several versions of both data entities and action entities have to be able to co-exist. When a recent action entity instance encounters an older data entity instance, it must interpret its data fields in the way the older action entity instance would have. If necessary, default values need to be defined for fields not present in the older data entity instance. When a more recent data entity instance is processed by an older action entity instance, only old data fields are used, because the older action entity instance is not aware of the recently added data field(s). To enable proper interaction with instances of older versions, or at least prevent version conflicts, instance version IDs are required (Theorems 5 and 6).

For example, suppose that in the motor control state machine example in Figures 3 and 6, we have an action entity version 2 (class version), which should process a data entity instance version 1 (data type). After reading the data entity instance's version ID, the action entity decides to never manipulate the 'FuseOK' data field, nor allows any transition to the state 'Failure'. These actions must be prevented because these fields do not exist in version 1 of the data entity; undefined behaviour would result. Instead, any information on the fuse or thermal cut out (if available) is ignored, corresponding to the (older) functionality of action entity version 1. Conversely, consider an instance of action entity version 1, which should process a younger instance of data entity version 2. This action entity instance is not even aware of the existence of fuse information nor the state failure, so it will never read nor manipulate these fields.

The potential 'ThermalCutout' transition of version 3 (Figure 7) from the 'On' state to the 'Failure' state will simply never happen if not both the data entity instance and action entity are of version 3. In addition, the action entity must include a selection to decide whether or not a reset command from the operator is needed for the transition from the 'Failure' state to the 'Off' state.

S4. *States or transitions shall not be deleted between versions.*

As explained in the previous section, it is a general rule in NST that deletions should not be allowed. Due to its importance, this is reiterated as a separate rule for state machines.

S5. *State modifications shall apply transparent coding or version wrapping.*

Deletions of states can cause violations of the version transparency theorems, so only additions and modifications are allowed. Modifications shall adhere to the principles of transparent coding or version wrapping as discussed in Section V-B.

S6.a *Entry actions of states will be implemented in a separate action entity.*

S6.b *Exit actions of states will be implemented in a separate action entity.*

S6.c *Transition effects will be implemented in a separate action entity.*

Entry actions, exit actions and transition effects determine the output of a state transition. The desired output of a transition can evolve separately from the design of the state machine itself. The three different types of actions resulting from a transition can also evolve independently, even when resulting from the same transition. According to the separation of concerns principle, they should therefore be separated.

VII. Implementation considerations

The architecture introduced in Section III-A is excellently suited to supporting co-existing, diverse versions of an evolvable state machine. For example, considering the scenario shown in Figure 5, motor 1 could be controlled by an instance of version 1 of the state machine presented in Section II, since status feedback is not required for it. Motor 2 is controlled by an instance of version 2 of this state machine, since for it the fuse condition must be taken into account. Motor 5 is also controlled by an instance of version 2 of the state machine; while motor 2 and motor 5 thus share the same instance version, they have different instance IDs.

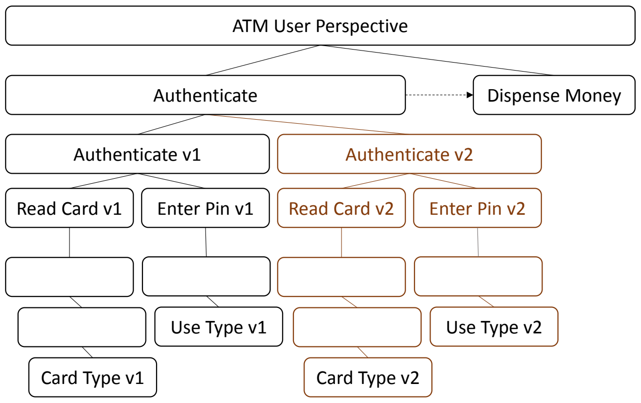


Figure 14. Non-evolvable implementation of the ATM

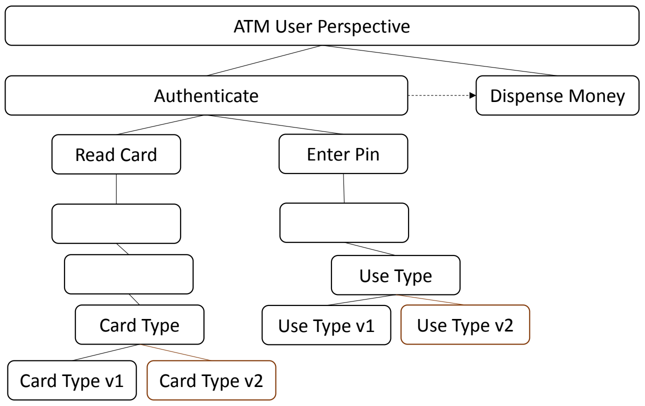


Figure 15. Evolvable implementation of the ATM

A similar approach can be used for the creation of evolvable information systems, which typically have a hierarchical nature. The use of hierarchical state machines makes the situation more complex, especially with respect to the implementation of version transparency. According to NST, changes must remain local. If a single change requires making changes in multiple places, the system is not evolvable.

Figure 14 shows the ATM example, with the different state machines represented in a hierarchy. The top level state machine contains two state machines 'Authenticate' and 'Dispense Money', the first of which contains the state machines 'Read Card' and 'Enter PIN'. To evolve from version 1 to version 2, both state machines need to be adapted. Hence, because of a single change in the requirements, two state machines need to be adapted. Version 1 of 'Enter PIN' can only be used with version 1 of 'Read Card', and version 2 of 'Enter PIN' only with version 2 of 'Read Card'. One could therefore decide to create two versions of the 'Authenticate' machine, each grouping a consistent pair of state machines.

This implementation would not yield the desired results.

The generic implementation of 'Authenticate' cannot decide which version needs to be used. Hence, it cannot be implemented using version wrapping or transparent coding. Moreover, this implementation is not evolvable according to NST. Both 'Read Card' and 'Enter PIN' are hierarchical state machines themselves, each comprising multiple levels of state machines. The change in 'Read Card' would be implemented by changing a state machine somewhere deep in the hierarchy, e.g., the 'Card Type' machine. The card type is consumed by the 'Use Type' machine somewhere in the hierarchy of the 'Enter PIN' machine. A single change would therefore imply the creation of new versions of all the state machines in the hierarchy, creating a non evolvable implementation.

Figure 15 shows an evolvable design where the changes remain local. New versions of the two machines that were changed are created. All other state machines are not impacted by the change. Assume that in the software implemented on an ATM, both version 1 and version 2 remain available. In that case 'Use Type' must be able to determine which version of 'Card Type' was used, as required by the version instance transparency law. This version cannot simply be passed through the hierarchy as this would again require the creation of new versions of all states machines in this hierarchy.

This can be solved by adapting the generic design of a state machine such that every state machine hierarchically enclosed in a state of a given state machine has access to the data element that contains the state and trigger information. The evolution from version 1 to version 2 would thus involve, apart from the evolution of the state machines themselves, a change to the data element containing the state of 'Authenticate' by adding a version number or the card type (or both).

VIII. Conclusion

The state machine is a valuable artefact for modelling systems. In a rapidly changing environment, there is a need for evolvable state machines. When production systems evolve, corresponding changes have to be made in the automation software; similarly, changes in a business environment may require the introduction of new states, new transitions or changes to existing transitions in one or more state machines used in an information system. However, when systems evolve, it follows from Lehman's law of increasing complexity that their further evolution is restrained when the systems' size increases over time.

State machines benefit from the introduction of modularity, as can be seen in the popularity of (hierarchical) UML statecharts. Normalized Systems Theory offers a theoretical foundation to achieve evolvability in modular structures by imposing restrictions on the definition of the modules and their interfaces.

This paper presented a design for evolvable state machines that can be used in both automation systems software and in information systems. The design is based on Normalized Systems Theory. Rules were derived to constrain changes to state machines in order to achieve the property of evolvability. In addition, case scenarios were discussed showing how instances of different versions of such an evolvable state machine can coexist.

The design supports dynamic reconfiguration, as called for by Kuhl and Fay, to update a system without the need for a complete system shutdown. In an automation system, compiling an IEC 61131-3 project includes allocating memory to variables. A shutdown is only necessary when this memory must be remapped. Changing the value of a data field in a data instance can be done without recompilation, so no shutdown is required. When, for example, a motor is replaced by a new one, this change is reflected by a change to the instance version ID of the data entity in our design. Therefore, dynamic reconfiguration is supported for such a situation. A similar conclusion holds for information systems in general if the restrictions imposed in this paper are adhered to.

Still, creating software that strictly adheres to the design rules proposed in this paper is a tedious, time consuming and error prone activity. This is, in fact, a major criticism often made in relation to Normalized Systems Theory. Many practitioners claim that it is not possible at all to create such software, except for small applications used for demonstration purposes only, i.e., that Lehman's law applies to all real world software systems.

From the beginning, this problem has been recognized as a major research question related to Normalized Systems Theory. Apart from the development of the theory, it has extensively been investigated how Normalized Systems compliant software can be built. For this purpose, a number of software patterns have been defined [36] [37], together with a set of pattern expanders that allow expanding a higher level description into NST compliant software. After an initial, academic, proof of concept, the development of the expanders became the core activity of the Normalized Systems Institute. The latter is a cooperation of the University of Antwerp with a number of industrial partners and government institutes. By now, the partners in the institute have developed a number of small and mid-sized information systems that are being used in a production environment, proving that it is possible to defeat Lehman's law in a real world software system.

Regarding future work, state machine libraries and toolkits should be improved by adding constraints to follow the rules presented in this paper, resulting in increased system evolvability by ensuring compliance with the theorems on Normalized Systems. The analysis still needs to be expanded to aspects of concurrency and the related UML constructs. Either would the rule set need to be revisited, or it needs to be demonstrated that real world information systems can be built based on state machines without these constructs.

References

[1] D. van der Linden, G. Neugschwandtner, and H. Mannaert, "Towards evolvable state machines for automation systems," in *Proc. 8th Intl. Conf. on Systems (ICONS)*, 2013, pp. 148–153.

[2] P. Stachour and D. Collier-Brown, "You don't know Jack about software maintenance," *Communications of the ACM*, vol. 52, no. 11, pp. 54–58, Nov. 2009.

[3] S. Ciraci and P. van den Broek, "Evolvability as a quality attribute of software architectures," in *Proc. 2nd Intl. ERCIM Workshop on Software Evolution*, 2006, pp. 29–31.

[4] I. Kuhl and A. Fay, "A middleware for software evolution of automation software," in *Proc. 16th IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA)*, 2011.

[5] (Retrieved in 2013) Atos Worldline, System stop. [Online]. Available: http://be.worldline.com/index/en_US/5253162/0000/System-stop.htm

[6] W. Stoecker, "Programmable controllers for industrial refrigerant plants," *International Journal of Refrigeration*, vol. 4, no. 6, pp. 329 – 334, 1981.

[7] E. I. Organick, *The Multics System: An Examination of Its Structure*. Cambridge, MA, USA: MIT Press, 1972.

[8] F. J. Corbató, J. H. Saltzer, and C. T. Clingen, "Multics: the first seven years," in *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, 1972, pp. 571–583.

[9] (Retrieved in 2013) Multics: Myths. [Online]. Available: http://www.multicians.org/myths.html

[10] O. Niggemann, "System-level design and simulation of automation systems," in *Proc. 8th IEEE Intl. Workshop on Factory Communication Systems (WFCS)*, 2010, pp. 173–176.

[11] IEC 61131-3, *Programmable controllers – Part 3: Programming languages*. International Electrotechnical Commission, 2003.

[12] R. H. Follett and J. E. Sammet, "Programming language standards," in *Encyclopedia of Computer Science*. John Wiley and Sons Ltd., 2003, pp. 1466–1470.

[13] (Retrieved in 2013) TIOBE Software, TIOBE Programming Community Index for September 2013. [Online]. Available: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[14] (Retrieved in 2013) ISO/IEC JTC1/SC22/WG21, C++-Standards. [Online]. Available: www.open-std.org/jtc1/sc22/wg21/docs/standards

[15] (Retrieved in 2013) The Apache Software Foundation, Status Of C++ 0x Language Features in Compilers. [Online]. Available: http://wiki.apache.org/stdcxx/C++0xCompilerSupport

[16] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 2006.

[17] ISA, *Batch Control – Part 1: Models and Terminology*. ANSI/ISA-88.01, 1995.

[18] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," in *Neurocomputing: foundations of research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 15–27.

[19] F. Wagner and P. Wolstenholme, "Misunderstandings about state machines," *IET Computing & Control Engineering Journal*, no. 4, Aug.–Sep. 2004.

[20] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, Jun. 1987.

[21] H. Fecher, J. Schönborn, M. Kyas, and W.-P. de Roever, "29 new unclarities in the semantics of UML 2.0 state machines," in *Proc. 7th Intl. Conf. on Formal Methods and Software Engineering*. Springer-Verlag, 2005, pp. 52–65.

[22] K. Lano, *UML 2 Semantics and Applications*. New York, NY, USA: John Wiley & Sons, 2009.

[23] S. Liu, Y. Liu, É. André, C. Choppy, J. Sun, B. Wadhwa, and J. S. Dong, "A formal semantics for the complete syntax of UML state machines with communications," in *Proc. 10th Intl. Conf. on Integrated Formal Methods (iFM'13)*, vol. 7940. Springer, 2013, pp. 331–346.

[24] D. L. Parnas, "Software aging," in *Proc. 16th Intl. Conf. on Software Engineering (ICSE)*, 1994, pp. 279–287.

[25] D. Van Nuffel, "Towards Designing Modular and Evolvable Business Processes," Ph.D. dissertation, University of Antwerp, 2011.

[26] M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, pp. 1060–1076, 1980.

[27] F. P. Brooks, Jr., *The Mythical Man-Month: Essays on Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1978.

[28] F. P. Brooks, Jr., "No silver bullet: Essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.

[29] D. van der Linden, G. Neugschwandtner, and H. Mannaert, "Industrial automation software: Using the Web as a design guide," in *Proc. 7th Intl. Conf. on Internet and Web Applications and Services (ICIW)*, 2012.

[30] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability," *Science of Computer Programming*, vol. 76, no. 12, pp. 1210–1222, 2011.

[31] I. Vanderfeesten, H. A. Reijers, and W. M. van der Aalst, “Evaluating workflow process designs using cohesion and coupling metrics,” *Computers in Industry*, vol. 59, no. 5, pp. 420–437, 2008.

[32] D. van der Linden and H. Mannaert, “In search of rules for evolvable and stateful run-time deployment of controllers in industrial automation systems,” in *Proc. 7th Intl. Conf. on Systems (ICONS)*, 2012, pp. 67–72.

[33] M. Snoeck and C. Michiels, “Domain modelling and the co-design of business rules in the telecommunication business area,” *Information Systems Frontiers*, vol. 4, no. 3, pp. 331–342, 2002.

[34] W. Lemahieu, M. Snoeck, F. Goethals, M. D. Backer, R. Haesen, J. Vandenbulcke, and G. Dedene, “Coordinating COTS applications via a business event layer,” *IEEE Software*, vol. 22, no. 4, pp. 28–35, 2005.

[35] D. van der Linden, H. Mannaert, W. Kastner, and H. Peremans, “Towards normalized connection elements in industrial automation,” *International Journal on Advances in Internet Technology*, vol. 4, no. 3&4, pp. 133–146, 2011.

[36] H. Mannaert, J. Verelst, and K. Ven, “Exploring concepts for deterministic software engineering: Service interfaces, pattern expansion, and stability,” in *Proc. IEEE Intl. Conf. on Software Engineering Advances (ICSEA)*, 2007.

[37] H. Mannaert and J. Verelst, *Normalized Systems: Re-creating Information Technology Based on Laws for Software Evolvability*. Koppa, 2009.

Sustainable Multiprocessor Real-Time Scheduling with Exact Preemption Cost

Falou Ndoye
INRIA Paris-Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
falou.ndoye@inria.fr

Yves Sorel
INRIA Paris-Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
yves.sorel@inria.fr

***Abstract*—In this paper, we address for safety critical applications the problem of multiprocessor real-time scheduling while taking into account the exact preemption cost. In the framework of multiprocessor real-time partitioned scheduling, we propose a greedy heuristic, which balances the load of the tasks on all the processors and minimizes the response time of the applications. That heuristic uses a schedulability condition, which is based on the $\oplus$ operation. That operation performs a schedulability analysis while taking into account the exact preemption cost. In this paper, the WCETs (Worst Case Execution Time) of tasks are considered rather than its EETs (Exact Execution Time). In this case, we prove that the schedulability analysis is sustainable. We also highlight the impact of the preemption cost in the schedulability analysis. A performance analysis is achieved, which compares the proposed heuristic to a branch and bound exact algorithm and to worst-fit and best-fit heuristics.**

Keywords-multiprocessor real-time scheduling; partitioned scheduling; exact preemption cost; sustainable; load balancing.

I. Introduction

For computation power and modularity issues, multiprocessor architectures are necessary to tackle complex applications found in domains such as avionics, automotive, rail, mobile robotics, etc. Some of these applications are safety critical [1], leading to hard real-time task systems whose number of resources are fixed and constraints must be necessarily satisfied in order to avoid catastrophic consequences. Although fixed priority preemptive real-time scheduling allows a better success ratio than non-preemptive real-time scheduling, preemption has a cost. That cost is usually approximated in the WCET (Worst Case Execution Time) as assumed, explicitly, by Liu and Layland in their pioneering article [2]. However, such approximation is dangerous in a safety critical context since an application could miss some deadlines during its real-time execution even though schedulability conditions have been satisfied. This is the reason why it is necessary to be aware of the exact preemption cost. In this paper, we address the problem of multiprocessor real-time scheduling while taking into account the exact preemption cost in the case of safety critical applications. In the framework of multiprocessor real-time partitioned scheduling, we propose a greedy heuristic [3], which balances the load of the tasks on all the processors and tends to minimize the response time of the tasks. That heuristic uses a schedulability condition based on the algebraic $\oplus$ operation, which performs a schedulability analysis taking into account the exact preemption cost.

The remainder of the paper is organized as follows: Section II presents related work about preemption cost and multiprocessor real-time scheduling. Section III describes the model and the schedulability analysis we use. Section IV presents the sustainability of this schedulability analysis. Section V presents the impact of the preemption cost in the schedulability analysis. Section VI presents the proposed multiprocessor scheduling heuristic as well as its complexity. Section VII presents a performance analysis that compares the heuristic with the Branch and Bound (B&B) exact algorithm and the Worst-Fit (WF) and Best-Fit (BF) heuristics. Finally, Section VIII concludes and gives some directions for future work.

II. Related work

A. Exact preemption cost in real-time scheduling

There have been very few studies addressing the exact number of preemptions. Among them, the most relevant are the following. A. Burns, K. Tindell and A. Wellings in [4] presented an analysis that enables the global cost due to preemptions to be factored into the standard equations for calculating the worst case response time of any task, but they achieved that by considering the maximum number of preemptions rather than the exact number. Juan Echagüe, I. Ripoll and A. Crespo also tried to solve the problem of the exact number of preemptions in [5] by computing the schedule using idle times and counting the number of preemptions. They did not take into account the cost of each preemption during the analysis. Hence, this amounts to considering only the minimum number of preemptions because some preemptions are not considered: those due to the increase in the execution time of the task because of the cost of preemptions themselves.

In order to reduce the preemption cost and improve the schedulability of tasks, a lot of work has focused on limited-preemption policies; among these we can cite fixed priority scheduling with deferred preemption (FPSDP) also called cooperative scheduling [6] and fixed priority scheduling with a preemption threshold (FPSPT) [7], [8]. According to FPSDP, each job of a task is a sequence of sub-jobs, where sub-jobs are not preemptive. When a job is being executed, it can only be preempted between two consecutive sub-jobs. For FPSPT, each task is assigned a nominal and a threshold priority. A preemption will take place only if the preempting task has a nominal priority greater than the preemption threshold of the executing task. None of the previous works considers the exact number of preemptions. Nonetheless, that can affect the correct behavior of the system at run-time, or in any case leads to resources being wasted in terms of time and memory. It is not difficult to determine the constant cost of every preemption, which includes the context switch necessary to make the preemption possible together with the choice of the task with the highest priority. However, the exact number of preemptions is difficult to determine since it may vary according to every instance of a task. To our best knowledge there are only few studies that take into account the exact preemption cost in the schedulability conditions, except those presented in [9], [10]. The authors proposed a scheduling operation named $\oplus$ that performs a schedulability analysis while computing the exact number of preemptions. Since the principle of this operation is not usual we give a detailed presentation in Section III-B.

B. *Multiprocessor real-time scheduling*

The scheduling of real-time tasks on multiprocessor architectures can be achieved according to three main approaches: partitioned scheduling, global scheduling, and semi-partitioned scheduling.

In the partitioned scheduling approach [11], [12] the system of tasks is divided into a number of disjoint subsystems less than or equal to the number of processors in the multiprocessor architecture, and each of these subsystems is allocated to one processor. All the instances, or jobs, of a task are executed on the same processor and no migration is permitted. In this approach, it is necessary to choose a scheduling algorithm for every processor, possibly the same algorithm, and also an allocation algorithm. On the other hand, the allocation problem has been demonstrated to be NP-Hard [13]. This complexity is the main drawback of the partitioned scheduling approach.

Heuristics are considered to be the best suited solutions when the execution time is crucial as in the rapid prototyping phase of the design process. In the case of fixed priority scheduling and independent tasks, Davari and Dhall were the first to propose in [14] two preemptive scheduling algorithms RM-FF (Rate Monotonic First Fit) and RM-NF (Rate Monotonic Next Fit) to solve the multiprocessor real-time scheduling problem. In the proposed algorithms, the uniprocessor RM algorithm [2] is used to verify if a task is schedulable on a processor with respectively First-Fit (FF) and Next-Fit (NF) to solve the allocation problem. Another heuristic, RM-BF (Rate Monotonic Best Fit) was proposed in [15]. It makes it possible to minimize the remaining processor load ($1-U_{p_j}$), called the unutilized capacity of the processor p_j [16], where U_{p_j} is the load of the tasks on p_j. In contrast to RM-BF, RM-WF [15] (Rate Monotonic Worst Fit) maximizes the remaining processor load. All these approaches uses the classical Liu and Layland [2] model of tasks that assumes the preemption cost is approximated in the WCET. In order to tackle this problem [17] presents a first solution to take into account the exact preemption cost in multiprocessor real-time scheduling.

In the global scheduling approach [11], [12], a unique scheduling algorithm is applied globally for every processor of the multiprocessor architecture. All the ready tasks are in a unique queue shared by all the processors. In this queue, the m tasks with the highest priorities are selected to be executed on the m available processors. Besides preemptions, task migrations are permitted. The advantage of the global scheduling approach, is that it allows a better use of the processors. The main drawback of the global scheduling approach, is that each migration nowadays has a prohibitive cost.

In the semi-partitioned scheduling approach [18], [19], derived from the partitioned scheduling approach, each task is allocated to a specific processor as long as the total utilization of the processor does not exceed its schedulable bound. In this approach, some tasks can be portioned for their executions among multiple processors. During run-time scheduling, a portioned task is permitted to migrate among the allocated processors, while the partitioned tasks are executed on specific processors without any migration. The semi-partitioned scheduling approach allows a reduction of the number in migrations. But again, it is necessary to be aware that every migration has a cost.

C. *Our choices*

The cost of migrations in the global and semi-partitioned scheduling approaches leads us to choose the partitioned scheduling approach. In addition, since the partitioned scheduling approach amounts to transform the multiprocessor scheduling problem into several uniprocessor scheduling problems, we can take advantage of the numerous research results obtained for the uniprocessor scheduling problem. In order to achieve rapid prototyping, we propose an allocation heuristic rather than a metaheuristic [20] or an exact algorithm [21], and a schedulability condition to verify if a task is schedulable on a specific processor. Next-fit (NF) and first-fit (FF) heuristics cannot optimize the load of the tasks on the processors, their choice is only based on the first processor, which satisfies the schedulability condition. The BF heuristic using the load as a cost function, tries to fill a processor as much as possible before using another one. This technique does not allow load balancing. The only heuristic among the bin-packing heuristics which permits load balancing is WF. But, as with all the other bin-packing heuristics, WF tries to reduce the number of processors and that limits the balancing while multiprocessor architectures used in industrial applications, which we are interested in, have a fixed number of processors. That is the reason why we propose a greedy heuristic similar to the WF heuristic, but which uses all the available processors. This heuristic aims at minimizing the load on each processor. That allows in turn the balance of the load on all the processors.

Preemptive scheduling algorithms are able to successfully schedule some task systems that cannot be scheduled by non-preemptive scheduling algorithms, but the preemption has a cost. Indeed, Liu and Layland [2] assume that the preemption cost is approximated in the WCET. Thus, there are two possible cases: the approximation in time and memory space is high enough and thus will probably lead to wasting resources, or the approximation is low and thus a task system declared schedulable may miss some deadlines during its execution at runtime. In order to take into account the exact preemption cost, we propose to use the $\oplus$ operation [9], [10]. This is an algebraic operation verifies if two tasks are schedulable, or not, while taking into account the exact preemption cost.

III. Model and Schedulability analysis

A. Model

Let $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$ be a system of n preemptive, independent and periodic real-time tasks. Every task is denoted by $\tau_i = (r_i^1, C_i, D_i, T_i)$ where r_i^1, C_i, D_i and T_i are the characteristics of the task. r_i^1 is the first activation date, C_i is the WCET without any approximation of the preemption cost, D_i is the relative deadline, and T_i the period of the task τ_i. We assume that $C_i \leq D_i \leq T_i$. Here we use the WCET rather than the EET (Exact Execution Time) used in [1] and we will prove that the schedulabity analysis is sustainable. We assume that Γ_n is scheduled off-line on m identical processors (all the processors have the same computation power). We assume the tasks have fixe priorities. Eventually, we assume that the processors have neither cache nor pipeline, or complex internal architecture. The latter assumption is usually made in safety critical applications where determinism is a key issue.

B. Schedulability analysis based on the $\oplus$ operation

Our schedulability analysis uses the $\oplus$ scheduling operation [10] . This operation is applied to a pair of tasks (τ_i, τ_j), such that τ_i has the highest priority. It gives as a result a task $\mathcal{R}$, that is $\mathcal{R} = \tau_i \oplus \tau_j$. The $\oplus$ takes into account the exact preemption cost incurred by the task τ_j. The schedulability interval, i.e., the interval in which we study the schedulability of the tasks, comes from the Theorem 1 below, which was given by J. Gossens [22].

Theorem 1: For a system $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$ of n periodic tasks arranged by decreasing priorities with respect to a fixed-priority scheduling policy, let $(s_i^{'})_{i \in \mathbb{N}^*}$ be the sequence defined by:

$$\begin{cases} s_1^{'} = r_1^1 \\ s_i^{'} = r_i^1 + \left\lceil \dfrac{(s_{i-1} - r_i^1)^+}{T_i} \right\rceil \cdot T_i, \quad 2 \leq i \leq n \end{cases} \quad (1)$$

If there exists a valid schedule of Γ_n until the time $s_n^{'} + H_n$ where $H_n = lcm\{T_i \,|\, i = 1, \cdots, n\}$, and $x^+ = max(x, 0)$, then this schedule is valid and periodic of period H_n from $s_n^{'}$.

A direct consequence of the previous theorem is that in the case of a valid schedule, the result of the schedule of the i first tasks is periodic of period $H_i = lcm\{T_j \,|\, j = 1, \cdots, i\}$ from $s_i^{'}$. Thus, the interval which precedes $s_i^{'}$ necessarily contains the *transient phase*, corresponding to the initial part of the schedule and the interval starting at time $s_i^{'}$ with length H_i is isomorphic to the *permanent phase* of the schedule of the i first tasks, which repeats identically from the instant $s_i^{'}$.

In order to compute $\mathcal{R} = \tau_i \oplus \tau_j$ with $j = i + 1$, we set $\epsilon = min(r_i^1, r_j^1)$. Since ϵ always exists, the interval $[\epsilon, s_j^{'}]$ defines the transient phase and the interval $[s_j^{'}, s_j^{'} + H_j]$ defines the permanent phase, s_j and H_j are given by the theorem 1. The schedulability study of the tasks is performed in the interval $[\epsilon, s_j^{'} + H_j]$. In this interval, the number of instances of a task τ_j is given by $n_j = \frac{(s_j^{'} + H_j) - r_j^1}{T_j}$.

1) Principle of the $\oplus$ operation: The principle of $\oplus$ applied to a pair of tasks (τ_i, τ_j) consists in replacing the available time units of the highest priority task τ_i with the time units of the lowest priority task τ_j. In order to do that, both tasks are initially referenced to the same time origin ϵ. Then, task τ_i is rewritten according to the number of instances of task τ_j in the interval $[r_j^1, s_j^{'} + H_j]$ of both task periods. This operation allows not only the identification of the available time units in task τ_i, but also the verification that task τ_j does not miss any deadlines.

When the task τ_j is preempted by the task τ_i the exact number of preemptions must be computed for each instance of τ_j by considering all its time units. When τ_j is preempted, we increment its number of preemptions and we add the cost associated with one preemption in the remaining execution of τ_j, i.e., the number of time units that τ_j must execute in order to complete its execution. That scheme is repeated to take into account a preemption generated by a previous preemption, and so on. In contrast to other works presented in the literature, this principle makes it possible to compute the exact number of preemptions. The cost associated to that exact number of preemptions is added to the WCET of τ_j to obtain its PET (Preemption Execution Time), i.e., the execution time taking into account the exact preemption cost.

Figure 1 illustrates the PET. In this figure, the PET of task τ_i in the instance $k+1$ is given by $C_i^{k+1} = C_i + 2\alpha$ due to two preemptions, with α being the cost of one preemption. If the amount of PET unit of times fits in the available time units of task τ_i, the task τ_j is schedulable, giving as a result task $\mathcal{R}$, otherwise it is not schedulable. $\oplus$ is an internal operation, i.e., the result given by $\oplus$ is also a task, that result may be in turn used as the highest priority task in another $\oplus$ operation. Thanks to this property it is possible to consider more than two tasks.

In order to perform the schedulability analysis of the task system $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$, ordered according to the decreasing priorities of the tasks, the $\oplus$ operation is applied from the task with the highest priority to the task with the lowest priority. Consequently, if $\mathcal{R}_n$ is the scheduling task result of Γ_n, then $\mathcal{R}_n$ is obtained by successive iterations:

$$\begin{cases} \mathcal{R}_1 = \tau_1 \\ \mathcal{R}_{i+1} = \mathcal{R}_i \oplus \tau_{i+1}, \quad 1 \leq i < n \end{cases}$$

As such we have $\mathcal{R}_n = ((\tau_1 \oplus \tau_2) \oplus \cdots \oplus \tau_{n-1}) \oplus \tau_n$. The system Γ_n will be said schedulable if and only if all the tasks are schedulable. If this is not the case, then the system Γ_n is said to be not schedulable.

The complexity of $\oplus$ applied to a pair of tasks τ_i and τ_j is $O(l)$ with l is the LCM between the period of τ_i and the period of τ_j.

We denote by $H_j = lcm\{T_l : \tau_l \in hp(\tau_j)\}$ where T_l represents the period of task τ_l and $hp(\tau_j)$ denotes the subsystem of tasks with a priority higher than the priority of τ_j. The number of instances of task τ_i in the permanent phase is given by:

$$\sigma_{perm_j} = \frac{H_j}{T_j} = \frac{lcm\{T_l : \tau_l \in hp(\tau_j)\}}{T_j} \tag{2}$$

The exact permanent load of a task τ_j, i.e., the load of the task τ_j, while taking into account the exact preemption cost, is given by:

$$U_j^* = \frac{C_j^*}{T_j} \text{ with } C_j^* = \frac{1}{\sigma_{perm_j}} \sum_{l=1}^{\sigma_{perm_j}} C_j^l \tag{3}$$

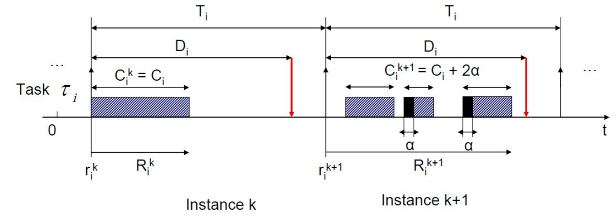


Figure 1. PET of a task

In (3), C_j^l corresponds to the PET of the l^{th} instance in the permanent phase. As such, the exact permanent load of the system Γ_n composed of n periodic tasks scheduled on a processor p_i is given by:

$$U_{p_i}^* = \sum_{j=1}^{n} U_j^* \tag{4}$$

2) Example: We apply the $\oplus$ operation to a system of periodic preemptive real-time tasks while taking into account the exact preemption cost. Let us consider such a system $\Gamma_3 = \{\tau_1, \tau_2, \tau_3\}$ of 3 tasks where τ_1 is the task with the highest priority, and τ_3 is the task with the lowest priority. We consider the cost of one preemption to be one time unit for all tasks. The characteristics of the tasks are summarized in Table I.

The $\oplus$ operation is applied to a pair of operands. The left operand called the "executed task" corresponds to the result of the tasks previously scheduled, and the right operand called the "executable task" corresponds to the task to be scheduled. We represent an instance of the executable task by a unique sequence of symbols "**e**", in bold, followed by a sequence of symbols "a". Each symbol "**e**", in bold, in the executable task represents an executable time unit, i.e., the time unit that the task to be scheduled, must execute. Each symbol "a" represents an available time unit. Actually, such representation is repeated indefinitely since the task is periodic. We represent an instance of an executed task by a sequence of symbols "e" followed by a sequence of symbols "a", possibly repeated several times. Each symbol "e" in the executed task represents one executed time unit, i.e., the time unit executed by all the tasks previously scheduled. From the end of the transient phase, given by theorem 1, such representation is repeated according to the LCM of the tasks already scheduled.

The $\oplus$ operation aims at replacing all the available time units of the executed task (left operand) by the executable time units of the executable task (right operand). In order to make both tasks comparable, first the executable task is repeated according to the number of its instances in the schedulabity interval. Second, the executed task is rewritten according to the number of instances of the executable task in the schedulability interval. Therefore, the task resulting of the $\oplus$ operation applied to a pair of tasks, is an executed task represented by a sequence of symbols "e" followed by a sequence of symbols "a", possibly repeated several times.

Table I
Tasks' characteristics

Tasks	r_i^1	C_i	D_i	T_i
τ_1	0	3	7	15
τ_2	5	2	6	6
τ_3	3	4	10	10

According to these definitions, each task instance of the system Γ_3 is represented as:

$$\begin{cases} \tau_1 = \{\mathbf{e}, \mathbf{e}, \mathbf{e}, a, a, a, a, a, a, a, a, a, a, a, a\} \\ \tau_2 = \{\mathbf{e}, \mathbf{e}, a, a, a, a\} \\ \tau_3 = \{\mathbf{e}, \mathbf{e}, \mathbf{e}, \mathbf{e}, a, a, a, a, a, a\} \end{cases}$$

The scheduling task result $\mathcal{R}_3$ that describes the schedule of the task system is obtained by the following successive iterations:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus \tau_1 \\ \mathcal{R}_i = \mathcal{R}_{i-1} \oplus \tau_i, \quad i = 2, 3 \end{cases}$$

Λ represents a task only composed with symbols "a" since there are no executed time units. $\mathcal{R}_1 = \Lambda \oplus \tau_1$ is computed as follows: according to equation (1) we have $s_1^{'} = 0$ and $H_1 = T_1 = 15$. Thus, the result of $\oplus$ applied to the pair (Λ, τ_1) is periodic of period $H_1 = T_1$ and is repeated indefinitely from $s_1^{'}$. We obtain $\mathcal{R}_1$ by replacing the 3 first available time units of Λ by the 3 executable time units of τ_1. Then, we have:

$$\mathcal{R}_1 = \{e, e, e, a, a, a, a, a, a, a, a, a, a, a, a\}_{[0,15]}$$

First iteration: Computation of $\mathcal{R}_2 = \mathcal{R}_1 \oplus \tau_2$. Thanks to equation (1), we have:

$$\begin{cases} s_1^{'} = 0 \\ s_2^{'} = 5 + \left\lceil \frac{(0-5)^+}{6} \right\rceil \cdot 6 = 5 \end{cases}$$

We have $H_2 = lcm(15, 6) = 30$, thus the transient phase belongs to the interval $[0, 5]$ and the permanent phase belongs to the interval $[5, 35]$. In the schedulability interval $[0, 35]$, $\mathcal{R}_1$ is rewritten as follows:

$$\mathcal{R}_1 = \{e, e, e, a, a\}_{[0,5]}\{a, a, a, a, a, a, a, a, a, a, e, e, e, a, a, a, a, a, a, a, a, a, a, a, a, e, e, e, a, a\}_{[5,35]}$$

Task τ_2 begins its execution at $t = 5$ corresponding to the beginning of the permanent phase. Its number of instances in the schedulability interval is $n_2 = \frac{(s_2^{'}+H_2)-r_2^1}{T_2} = \frac{(5+30)-5}{6} = \frac{30}{6} = 5$. According to the number of instances of τ_2 in the schedulability interval, $\mathcal{R}_1$ is rewritten as follows:

$$\begin{aligned} &\mathcal{R}_1 = \{e, e, e, a, a\}_{[0,5]}\{a, a, a, a, a, a\} \\ &\{a, a, a, a, e, e, \}\{e, a, a, a, a, a\} \\ &\{a, a, a, a, a, a\}\{a, e, e, e, a, a\} \end{aligned} \qquad (5)$$

$\mathcal{R}_2 = \mathcal{R}_1 \oplus \tau_2$ is obtained by replacing in the equation (5) for each corresponding instance of τ_2 in $\mathcal{R}_1$, the available time units "a" of $\mathcal{R}_1$ with the executable time units "**e**", in bold, of τ_2. During this replacement a preemption of τ_2 by τ_1 corresponds to the transition ("a" $\rightarrow$ "e"). The preemption of τ_2 by τ_1 is denoted by the time unit "**p**" called preemption time unit. When τ_2 is preempted, the next available time unit of $\mathcal{R}_1$ after this preemption is replaced by a preemption time unit "**p**". After replacing all the available time units of τ_1 with the executable time units of τ_2 and after adding the preemption time unit "**p**" in $\mathcal{R}_1$, we obtain:

$$\begin{aligned} &\mathcal{R}_2 = \{e, e, e, a, a\}_{(0,5)}\{\mathbf{e}, \mathbf{e}, a, a, a, a\} \\ &\{\mathbf{e}, \mathbf{e}, a, a, e, e\}\{e, \mathbf{e}, \mathbf{e}, a, a, a\} \\ &\{\mathbf{e}, \mathbf{e}, a, a, a, a\}\{\mathbf{e}, e, e, e, \mathbf{p}, \mathbf{e}\} \end{aligned}$$

For each corresponding instance of τ_2 in $\mathcal{R}_2$, its PET is given by the sum of the number of its executable time units **e**, in bold, and the number of its preemptions time unit "**p**". In the 4 first corresponding instances of τ_2 in $\mathcal{R}_2$, the PETs are the same and equal to 2 (PET=WCET) because τ_2 is not preempted in these instances, but in its 5^{th} instance, it is preempted once. That is the reason why its PET is equal to 3. In any corresponding instance of τ_2 in $\mathcal{R}_2$, the PET fits in the available time units left by $\mathcal{R}_1$ in this instance. Thus, the task τ_2 is schedulable while taking into account the exact preemption cost. Actually, we have:

$$\mathcal{R}_2 = \{e, e, e, a, a\}_{[0,5]}\{e, e, a, a, a, a, e, e, a, a, e, e, e, e, e, a, a, a, e, e, a, a, a, a, e, e, e, e, p, e\}_{[5,35]}$$

The differences with the previous expression of $\mathcal{R}_2$ is that the executable time units "**e**", in bold, become executed time units "e", and $\mathcal{R}_2$ does not exhibit the corresponding instances of τ_2.

Second iteration: Computation of $\mathcal{R}_3 = \mathcal{R}_2 \oplus \tau_3$. Thanks to equation (1), we have:

$$\begin{cases} s_2^{'} = 5 \\ s_3^{'} = 3 + \left\lceil \frac{(5-3)^+}{10} \right\rceil \cdot 10 = 13 \end{cases}$$

We have $H_3 = lcm(lcm(15, 6), 10) = lcm(30, 10) = 30$, thus the transient phase belongs to the interval $[0, 13]$ and the permanent phase belongs to the interval $[13, 43]$. In the schedulability interval $[0, 43]$, $\mathcal{R}_2$ is rewritten as follows:

$$\mathcal{R}_2 = \{e, e, e, a, a, e, e, a, a, a, a, e, e\}_{[0,13]}\{a, a, e, e, e, e, e, a, a, a, e, e, a, a, a, a, e, e, e, e, p, e, e, e, a, a, a, a, e, e\}_{[13,43]}$$

Task τ_3 begins its execution during the transient phase at $t = 3$. Its number of instances in the schedulability interval is $n_3 = \frac{(s_3^{'}+H_3)-r_3^1}{T_3} = \frac{(13+30)-3}{10} = \frac{40}{10} = 4$. According to the number of instances of τ_3 in the schedulability interval, $\mathcal{R}_2$ is rewritten as follows:

$$\begin{aligned} &\mathcal{R}_2 = \{e, e, e\}\{a, a, e, e, a, a, a, a, e, e\}_{[3,13]} \\ &\{a, a, e, e, e, e, e, a, a, a\}\ \{e, e, a, a, a, a, e, e, \\ &e, e\}\{p, e, e, e, a, a, a, a, e, e\} \end{aligned} \qquad (6)$$

$\mathcal{R}_3 = \mathcal{R}_2 \oplus \tau_3$ is obtained by replacing in the equation (6) for each corresponding instance of τ_3 in $\mathcal{R}_2$, the available time units "a" of $\mathcal{R}_2$ with the executable time units "**e**", in bold, of τ_3. During this replacement, a preemption of the task τ_3 by τ_1 or by τ_2 corresponds to a transition ("a" $\rightarrow$ "e"). When τ_3 is preempted, the next available time unit of $\mathcal{R}_2$ is replaced by a preemption time unit "**p**". After replacing the available time units of $\mathcal{R}_2$ with the executable time units of τ_3 and after adding the preemption time units "**p**" in $\mathcal{R}_2$, we obtain:

$$\mathcal{R}_3 = \{e,e,e\}\{\mathbf{e},\mathbf{e},e,e,\mathbf{p},\mathbf{e},\mathbf{e},a,e,e\}_{[3,13]}$$
$$\{\mathbf{e},\mathbf{e},e,e,e,e,e,\mathbf{p},\mathbf{e},\mathbf{e}\}\{e,e,\mathbf{e},\mathbf{e},\mathbf{e},\mathbf{e},e,e,e,e\}$$
$$\{p,e,e,e,\mathbf{e},\mathbf{e},\mathbf{e},\mathbf{e},e,e\}$$

For each corresponding instance of τ_3 in $\mathcal{R}_3$, its PET is given by the sum of the number of its executable time units "**e**", in bold, and the number of its preemptions time units "**p**". In the 2 first corresponding instances of τ_3 in $\mathcal{R}_3$, the task τ_3 suffers one preemption. Its PETs in every instance are the same and equal to 5. In its other instances there is no preemption of τ_3 and the PETs of τ_3 in these instances are the same and equal to 4 (PET=WCET). In any corresponding instance of τ_3 in $\mathcal{R}_3$, the PET fits in the available time units of $\mathcal{R}_2$ in this instance. Thus, the task τ_3 is schedulable while taking into account the exact preemption cost. Finally, we have:

$$\mathcal{R}_3 = \{e,e,e,e,e,e,e,p,e,e,a,e,e\}_{[0,13]}\{,e,e,e,$$
$$e,e,e,e,p,e,e,e,e,e,e,e,e,e,e,e,e,p,e,e,e,e,e,$$
$$e,e,e,e\}_{[13,43]}$$

The differences with the previous expression of $\mathcal{R}_3$ is that the executable time units "**e**", in bold, become executed time units "e", and $\mathcal{R}_3$ does not exhibit the corresponding instances of τ_3.

Since all the tasks are schedulable then the system $\Gamma_3 = \{\tau_1, \tau_2, \tau_3\}$ is schedulable.

Figure 2 presents the result of the schedule of Γ_3. In this figure, the permanent phase corresponds to the highlighted zone of the schedule and the transient phase corresponds to the interval preceding that zone. The disk represents only the permanent phase in a more compact form. This double representation of the schedule is obtained from the SAS software [23].

IV. SUSTAINABILITY

We consider the WCETs of the tasks instead of its EETs used in [1]. Thus, we have to show that if the execution time of a task at runtime is smaller than its WCET, the tasks system remains schedulable.

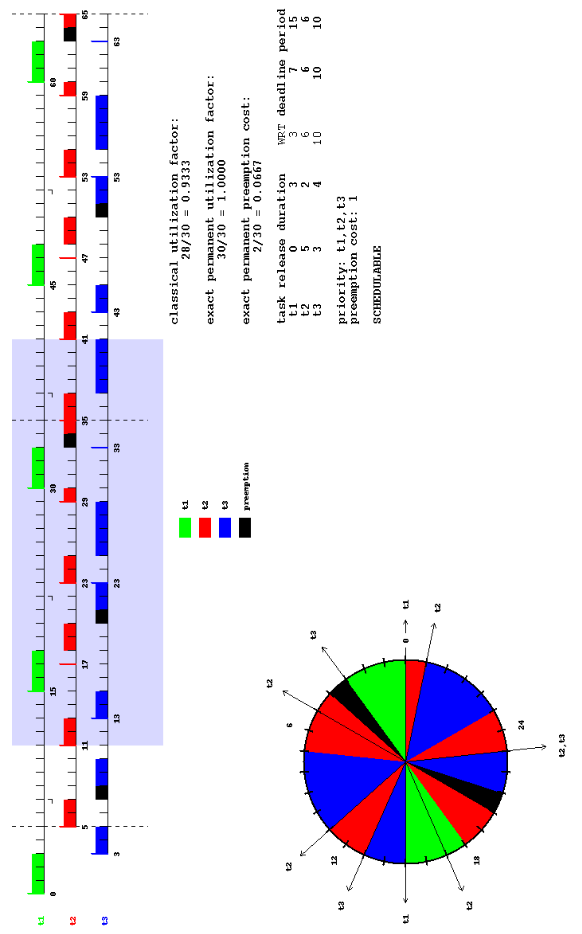
Definition 1: A schedulability analysis is sustainable [24] if any system considered to be schedulable according to this schedulability analysis remains schedulable when the parameters of one or more tasks change in any, some, or all the following way: 1) decreased execution time, 2) increased period, 3) increased deadline.

In this paper, we consider for the sustainability only the first property since we assume that the periods and the deadlines of the tasks are fixed.

Theorem 2: $\Gamma_n = \{\tau_1, \cdots, \tau_i, \cdots, \tau_n\}$, a system of independent periodic tasks arranged by decreasing priorities with respect to the RM fixed-priority scheduling policy, with $\tau_i = (r_i^1, C_i, D_i, T_i)$.

If Γ_n is schedulable according to $\oplus$ operation, then $\Gamma_n^{'} = \{\tau_1, \cdots, \tau_i^{'}, \cdots, \tau_n\}$, with $\tau_i^{'} = (r_i^1, C_i^{'}, D_i, T_i)$ and $C_i^{'} < C_i$, is also schedulable according to $\oplus$.

Proof: Assume that $\Gamma_n = \{\tau_1, \cdots, \tau_i, \cdots, \tau_n\}$ is schedulable while taking into account the exact preemption cost. We will show by contradiction that $\Gamma_n^{'} = \{\tau_1, \cdots, \tau_i^{'}, \cdots, \tau_n\}$ is also schedulable. For that, we assume that $\Gamma_n^{'}$ is not schedulable while taking into account the exact preemption preemption. That means that $\exists \tau_l \in \Gamma_n^{'}$ such that τ_l is not schedulable. Since the $i-1$ first tasks in $\Gamma_n^{'}$ are the highest priority tasks and are the same as in Γ_n



```
classical utilization factor:
        28/30 = 0.9333

exact permanent utilization factor:
        30/30 = 1.0000

exact permanent preemption cost:
        2/30 = 0.0667

task release duration   WRT deadline period
 t1     0        3       3      7      15
 t2     5        2       6      6       6
 t3     3        4      10     10      10

priority: t1,t2,t3
preemption cost: 1

SCHEDULABLE
```

Figure 2. Result of the scheduling of Γ_3, taking into account the exact preemption cost

then their schedule does not change then $i \leq l \leq n$.

Assume that $i = l$, then task $\tau_i^{'}$ is not schedulable. That means that there is an instance k of $\tau_i^{'}$ in which task $\tau_i^{'}$ cannot replace all its $C_i^{'k}$ executable time units by the available time units after the execution of the $i-1$ first tasks. Since we have independent tasks with static priorities and the tasks $\tau_i^{'}$ and τ_i have the same priority so if $C_i^{'} \leq C_i$ then the $C_i^{'k} \leq C_i^k \; \forall k \geq 1$. Thus, there exists also an instance k of τ_i in which τ_i is not schedulable, meaning that Γ_n is not schedulable. That is in contradiction according to our assumption that Γ_n is schedulable. Similarly if $l > i$ then Γ_n is not schedulable. Thus, if $\Gamma_n^{'}$ is not schedulable then Γ_n is not schedulable. That is equivalent to if Γ_n is schedulable then $\Gamma_n^{'}$ is schedulable. ■

V. Impact of the preemption cost in the schedulability analysis

In this section, we present the impact of the preemption cost in the schedulability analysis presented in the Section III-B. For that we consider two cases. In the first case, we assume that the preemption cost is approximated in the WCET of the tasks ($\alpha = 0$). In the second case, we assume that the preemption cost is not approximated into the WCET with a cost $\alpha = 1$ for one preemption. In order to show the impact of the preemption cost in the schedulability analysis, we compare the success ratio for these two cases. The success ratio is defined for a set of task systems, by:

$$\frac{number\ of\ task\ systems\ schedulable\ with\ \oplus}{number\ of\ task\ systems\ in\ a\ set\ of\ task\ systems}$$

For that, we generate randomly 15 sets of task systems. Every set of task systems is composed of 10 task systems. Every of the latter task systems contains 10 tasks. We compute the success ratio and the average load of every set of task systems.

Figure 3 shows that when the average load of the set of task systems is less than 0.8 then the scheduling, without and with preemption cost, has the same success ratio. But when this average load increases to 1 then the success ratio with preemption cost decreases until to be equal to 0, while the success ratio without preemption cost is decreases until to be equal to 0.8. When the average load is greater than 0.93, the success ratio with preemption cost is equal to 0, that is, no task system is schedulable whereas in the case without preemption cost some task sets are schedulable. This is the reason why preemption cost must be taken into account in safety critical systems.

VI. Multiprocessor scheduling heuristic

The heuristic presented in *Algorithm*1 is a greedy heuristic. The solution is built step by step. In each step a decision is taken and this decision is never questioned during the following steps (no backtracking). The effectiveness of such a greedy heuristic is based on the decision taken to build a new element of the solution. In our case, the decision is taken according to a cost function, which aims at minimizing the load.

A. Cost function

The cost function allows the selection of the best processor p_j to schedule a task τ_i. In our case, this cost function is the load $U_{p_j}^*$ (equation (4)) of the task τ_i and all the tasks already allocated on the processor p_j. The processor which minimizes this cost function for τ_i among all the processors, is considered to be the best processor to schedule the task τ_i.

In the case of the previous example, according to (4), the exact permanent load of the system Γ_3 scheduled on a processor p is given by:

$$U_p^* = \frac{3}{15} + \frac{1}{6} \cdot \frac{(2+2+2+2+3)}{5} + \frac{1}{10} \cdot \frac{(5+4+4)}{3} = 1$$

B. Principle of our allocation heuristic

We use a "list heuristic" [25]. In our case, we initialize this list, called the "candidate task system", with the task system given as input. We use for that candidate task system the decreasing order of the task priorities (according to RM fixed-priority scheduling policy [2]). At each step of the heuristic, the task with the highest priority is selected among the candidate task system, and we attempt to allocate it to its best processor according to the cost function presented previously. The heuristic minimizes the load $U_{p_j}^*$ of the task system on the different processors. It is similar to the WF bin-packing heuristic except that in the current iteration all the processors are used rather than only the processors visited during the previous iteration.

If Γ_n is the task system with n tasks and m is the number of processors, the complexity in the worst case of our heuristic is equal to $O(n.m.l)$, with $l = lcm\{T_i : \tau_i \in \Gamma_n\}$.

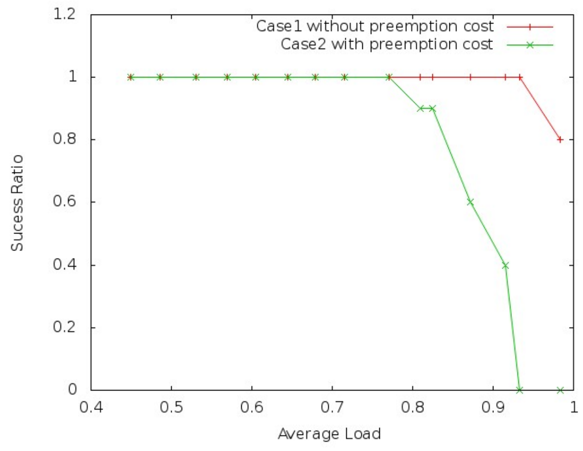


Figure 3. Impact of the preemption cost

Algorithm 1 Greedy heuristic

```
1: Initialize the candidate task system $W$ with the task system given as input and in the decreasing order of their priorities, initialize the boolean variable $TasksSchedulable$ to $true$
2: while $W$ is not empty and $TasksSchedulable = true$ do
3:     Select in $W$ the highest priority task $\tau_i$
4:     % Verify on each processor $p_j$ if task $\tau_i$ is schedulable.%
5:     for $j = 1$ to $m$ do
6:         if task $\tau_i$ is schedulable on $p_j$ with the exact preemption cost (scheduling operation $\oplus$ [10]) then
7:             Compute the cost function of task $\tau_i$ on the processor $p_j$, i.e., the load of $p_j$ using (4) given in Section III-B
8:         end if
9:     end for
10:    % Using the cost function again, choose the best processor for $\tau_i$ among all the processors on which $\tau_i$ is schedulable.%
11:    if $\tau_i$ is schedulable on one or several processors then
12:        Schedule the task $\tau_i$ on the processor which minimizes the cost function
13:        Remove the task $\tau_i$ from $W$.
14:        $TasksSchedulable = true$
15:    else
16:        $TasksSchedulable = false$
17:    end if
18: end while
```

The partitionned scheduling algorithm 1 is sustainable since we use on each processor a sustainable schedulibity analysis. That is means that, if the execution time of a task is smaller than its WCET, the tasks system remains schedulable.

VII. Performance Analysis

Our heuristic is compared with the B&B exact algorithm and the WF and BF heuristics. The B&B enumerates all the possible solutions in order to find the best solution, which minimizes the load of the tasks on the processors. In the B&B, WF and BF heuristics, we use the $\oplus$ operation presented in Section III-B as the schedulability condition. We compare the algorithms according to their execution time, their success ratio, the response time of the task systems, i.e., the total execution time of the tasks, and the unutilized capacity of the processors used during the allocation.

A. Execution time of the heuristics

We perform two kinds of tests to compare the execution time of the four algorithms. First, we fix the number of processors to 10 and we vary the number of tasks between 100 to 1000 tasks. Every task system is scheduled with the four algorithms and the corresponding execution times are computed. We obtained the results shown in Figure 4. In the second test, we use a single task system composed of 1000 tasks randomly generated and we vary the number of processors. We obtained the results shown in Figure 5.

In both tests, we notice that the exact algorithm explodes very quickly whereas the heuristics keep a reasonable execution time. Our heuristic up to 1000 tasks is close to the WF and BF heuristics in terms of execution time. However, for higher numbers of tasks less good results are obtained with our heuristic. In Figure 5 we also notice that when the number of processors varies, the execution times of WF and BF are constant, because these heuristics use the minimum number of processors. Another remark about Figure 5 is that the execution time of our heuristic does not increase monotonically with the number of processors, in contrast to Figure 4. Indeed, in our heuristic, increasing the number of processors leads to better balance the load of the tasks on all the processors. That increase in terms of processors, can decrease locally the LCM of the tasks on some processors, and consequently can reduce the execution time of the $\oplus$ operation.

B. Success ratio

In these tests, we compare the success ratio of our heuristic with the B&B exact algorithm and the WF and BF heuristics. The success ratio of an algorithm is defined as follows:

$$\frac{number\ of\ task\ systems\ schedulable\ with\ algorithm\ 1}{number\ of\ task\ systems\ in\ a\ set\ of\ task\ systems}$$

Due to the complexity of the B&B and in order to compare it with the heuristics, we executed each algorithm on 6

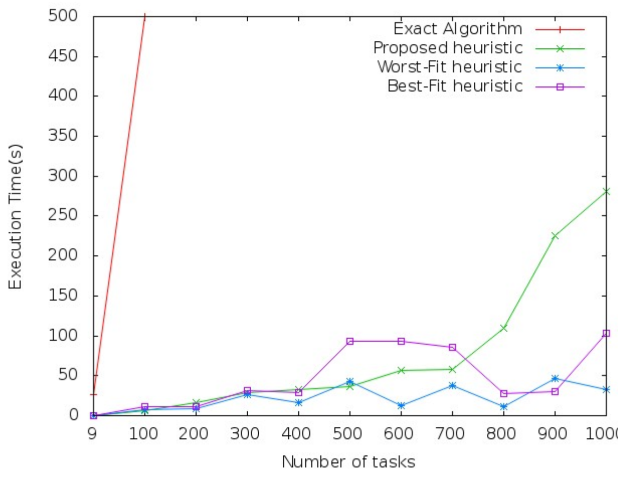


Figure 4. Execution time of the algorithms according the number of tasks

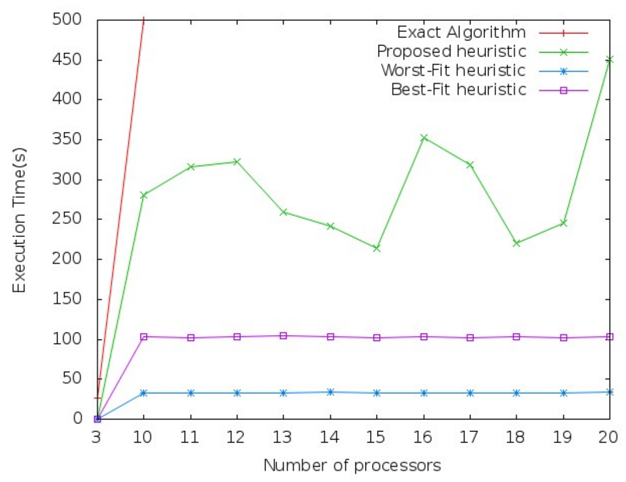


Figure 5. Execution time of the algorithms according to the number of processors

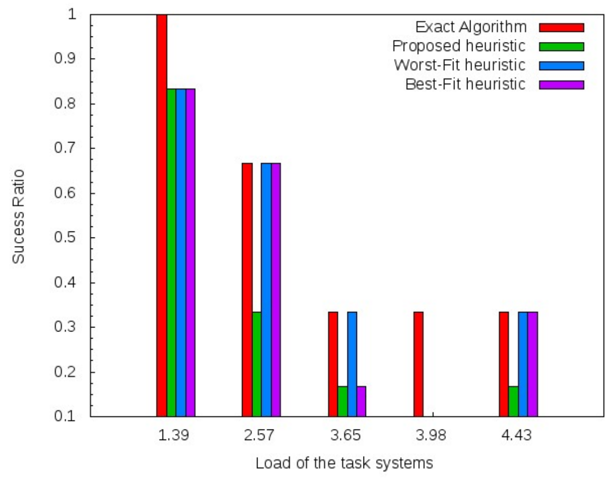


Figure 6. Sucess ratio

task systems. Each task system is composed at most of 10 randomly generated tasks and is executed on 2 processors. At each execution we determine for each algorithm the number of schedulable task systems.

As shown in Figure 6, we notice that WF and BF give better results than our heuristic in terms of success ratio. This loss in terms of success ratio is largely compensated by the gain in terms of response time of the task systems and by the unutilized capacity of the processors, as described in the Sections VII-C and VII-D.

C. Response time of the task systems

In these tests, we consider 10 task systems. The number of tasks in the task systems varies between 100 and 1000 randomly generated tasks and each task system is executed on 10 processors. We limit the tests to the WF and BF heuristics and our proposed heuristic because of the complexity of the B&B exact algorithm and we know that the B&B already gives better results than the heuristics. For each task system, we determine the allocation found by each heuristic and for this allocation the response time of the task system, i.e., the total execution time of all the tasks, is computed. We compare the response time of the task systems between the heuristics, as shown in Figure 7.

In this figure, we notice that the allocation found by our heuristic gives a better response time than those found by WF and BF. This is due to the fact that the execution of the tasks is parallelized on all the available processors whereas WF and BF attempts to reduce the number of processors rather than parallelize the execution of the tasks.

D. Average of the unutilized capacity of the processors

In these tests, we consider 10 task systems. The number of tasks in the task systems varies between 100 and 1000 randomly generated tasks and each task system is executed on 10 processors. We limit the tests to the WF and BF heuristics and our proposed heuristic because of the complexity of the B&B exact algorithm. In addition, we know that the B&B already gives better results than the heuristics. For each task system we determine the allocation found by each heuristic and for this allocation we compute the average of the remaining processor load $(1 - U_{p_j})$, called unutilized capacity, on the processors p_j used in this allocation. We compare the unutilized capacity of the processors used with the heuristics as shown in Figure 8.

In this figure, we observe that the allocation found by our heursitic gives for each processor more flexibility, i.e., more unutilized capacity, than those found by WF and BF. This is due to the fact that our heuristic balances the load of the tasks on all the processors, which ensures an execution time slack on each processor, whereas the BF heuristic fills the

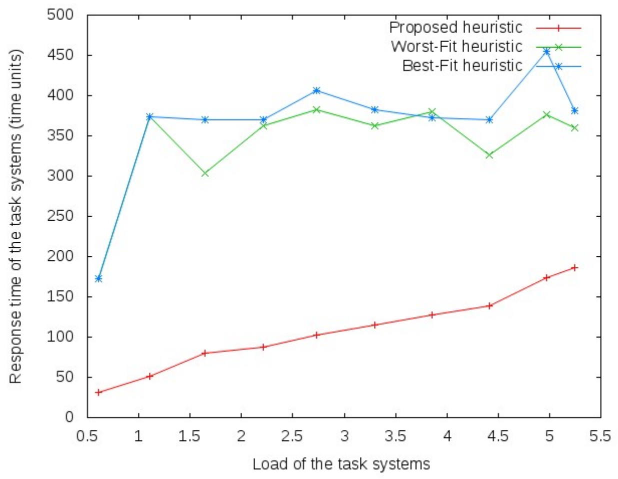


Figure 7. Execution time of the task systems

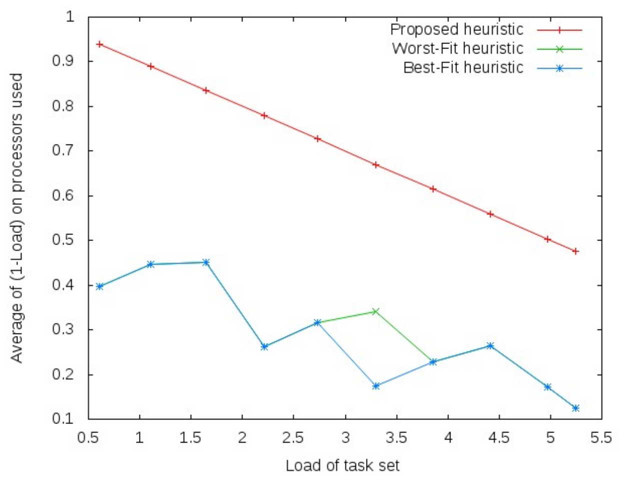


Figure 8. Average of (1-load) on the processors used

processors as much as possible. On the other hand, the WF heuristic, balances the load only on the processors already used and does not consider all the available processors.

VIII. Conclusion and future work

We have presented a greedy heuristic, which allocates and schedules, on a multiprocessor architecture, a system of real-time tasks while balancing the load on the processors. In addition, this heuristic takes into account the exact preemption cost that must be carefully considered in safety critical applications, which is the focus of our work. For that, we have used $\oplus$ operation as scheduling analysis and we have proved that $\oplus$ is sustainable, i.e., the task system remains schedulable at runtime when a task has an execution time smaller than its worst case execution time. We have also shown the impact of the preemption cost in the schedulability analysis of the safety critical systems.

We have carried out a performance analysis showing that, up to 1000 tasks, the proposed greedy heuristic, has results close to those of the WF and BF heuristics in terms of execution time. For higher number of tasks less good results can be obtained with our heuristic. On the other, hand the proposed heuristic is better than the WF and BF heuristic in terms of load balancing and flexibility, i.e., more unutilized capacity, of tasks at run-time. Also the allocation found with our heuristic has better response time than those with WF and BF heuristics.

In future works, we plan to study the multiprocessor real-time scheduling of dependent tasks, which leads to deal with data transfers and shared data management.

We have proved that the scheduling analysis using the $\oplus$ operation is sustainable, i.e., the task system remains schedulable at runtime when a task has an execution time smaller than its worst case execution time. We have also shown the impact of the preemption cost in the schedulability analysis of the safety critical systems.

References

[1] F. Ndoye and Y. Sorel. Safety critical multiprocessor real-time scheduling with exact preemption cost. In *the 8th International Conference on Systems, ICONS'13*, Seville, Espain, January 2013.

[2] C. L. Liu and J W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environnment. *Journal of the ACM*, vol. 20(1), January 1973.

[3] E.G. Coffman, G. Galambos, S. Martello, and Daniele Vigo. Bin packing approximation algorithms: Combinatorial analysis. *Handbook of combinatorial optimization*, 1998.

[4] A. Burns, K. Tindell, and A. Wellings. Effective analysis for engineering real-time fixed priority schedulers. *IEEE Trans. Softw. Eng.*, vol. 21, pp. 475-480, May 1995.

[5] J. Echague, I. Ripoll, and A. Crespo. Hard real-time preemptively scheduling with high context switch cost. In *Proceedings of 7th Euromicro workshop on Real-Time Systems*, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[6] Alan Burns. Preemptive priority-based scheduling: An appropriate engineering approach. *Advances in Real-Time Systems, chapter 10*, pp. 225–248, 1994.

[7] Y. Wang and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *Proceedings of the 6 International Conference on Real-Time Computing Systems and Applications*, RTCSA'99, Washington, DC, USA, 1999.

[8] M. Saksena and Y. Wang. Scalable real-time system design using preemption thresholds, November 2000.

[9] P. Meumeu Yomsi and Y. Sorel. Extending rate monotonic analysis with exact cost of preemptions for hard real-time systems. In *Proceedings of 19th Euromicro Conference on Real-Time Systems, ECRTS'07*, Pisa, Italy, July 2007.

[10] P. Meumeu Yomsi and Y. Sorel. An algebraic approach for fixed-priority scheduling of hard real-time systems with exact preemption cost. Research Report RR-7702, INRIA, August 2011.

[11] R.I. Davis and A. Burns. A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems. Technical report, University of York, Department of Computer Science, 2009.

[12] O.U.P. Zapata and P.M. Alvarez. Edf and rm multiprocessor scheduling algorithms: Survey and performance evaluation. Technical Report No. CINVESTAV-CS-RTG-02, CINVESTAV-IPN, Seccin de Computacin, Oct 2005.

[13] Garey and Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, NY, USA, 1979.

[14] S.K. Dhall and C.L. Liu. On a real-time scheduling problem. *Operation Research*, vol. 26(1), 1978.

[15] Y. Oh and S.H. Son. Tight performance bounds of heuristics for a real-time scheduling problem. Technical Report CS-93-24, Univ. of Virginia. Dep. of Computer Science, Charlottesville, VA 22903, May 1993.

[16] I. Lupu, P. Courbin, L. George, and J. Goossens. Multi-criteria evaluation of partitioning schemes for real-time systems. In *The 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'10*, Bilbao, Spain, September 2010.

[17] F. Ndoye and Y. Sorel. Preemptive multiprocessor real-time scheduling with exact preemption cost. In *Proceedings of 5th Junior Researcher Workshop on Real-Time Computing, JRWRTC'11, in conjunction with the 18th International conference on Real-Time and Network Systems, RTNS'11*, Nantes, France, September 2011.

[18] S. Katoa and N. Yammasaki. Semi-partitioning technique for multiprocessor real-time scheduling. In *Proceedings of WIP Session of the 29th Real-Time Systems Symposium (RTSS), IEEE Computer Society*, 2008.

[19] J. H. Anderson, V. Bud, and C. U. Devi. An edf-based scheduling algorithm for multiprocessor soft real-time systems. *In Proceedings of the 17th Euromicro Conference on Real-Time Systems*, pp. 199–208, Washington, DC, USA, 2005.

[20] E. G Talabi. *Metaheuristics*. Wiley, 2009.

[21] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *In Handbook of Applied Optimization*, pp. 65-67, Oxford University Press, 2002.

[22] J. Goossens. *Scheduling of Hard Real-Time Periodic Systems with Various Kinds of Deadline and Offset Constraints*. PhD thesis, Universit Libre de Bruxelles, 1999.

[23] P. Meumeu Yomsi, L. George, Y. Sorel, and D. de Rauglaudre. Improving the quality of control of periodic tasks scheduled by fp with an asynchronous approach. *International Journal on Advances in Systems and Measurements*, 2(2), 2009.

[24] S. Baruah and A. Burns. Sustainable scheduling analysis. *In Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pp. 159-168, Washington, DC, USA, 2006.

[25] L.T. Adams, K. M. Chandy, and J. R. Dickson. A comparison of list schedules for parallel processing systems. *Commun. ACM*, vol. 17, 685–690, December 1974.

Multilevel Flash Memories: Channel Modeling, Capacities and Optimal Coding Rates

Xiujie Huang*, Aleksandar Kavcic*, Xiao Ma†, Guiqiang Dong‡, and Tong Zhang§
*Dept. of Elect. Eng., University of Hawaii, Honolulu, HI 96822 USA
†Dept. of Elect. and Commun. Eng., Sun Yat-sen University, Guangzhou, GD 510006 China
‡Skyera Inc., San Jose, CA 95131 USA
§Dept. of Elect., Comp. and Syst. Eng., Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA
Email: {xiujie,kavcic}@hawaii.edu, maxiao@mail.sysu.edu.cn, dongguiqiang@gmail.com, tong.zhang@ieee.org

***Abstract*—This paper is concerned with channel modeling and capacity evaluation of the multilevel flash memory with m levels. The m-level flash memory is modeled as an m-amplitude-modulation channel with input-dependent additive Gaussian noise whose standard deviation depends on the channel input. The capacity as well as the optimal coding rate of an m-level flash memory channel (m-LFMC) is given. If the channel output is observed after a (finite) quantizer, then the channel is further transformed into a discrete memoryless channel, which yields an approximation of the capacity for the m-LFMC. Actually, the determination of the capacity for the m-LFMC can be transformed into a two-step optimization problem, which can be numerically solved by an alternating iterative algorithm. This algorithm delivers not only the optimal input/level distribution but also the optimal values of levels. This algorithm also delivers the optimized number of levels at any given voltage-to-deviation ratio. Numerical results are presented to show the consistency with well-known Smith's results for the amplitude-limited AWGN channel and the applicability of the modeling method, and to reveal that a finite level quantization of the channel output for the m-LFMC suffers from a negligible loss of information rate compared to the capacity.**

Keywords*—Amplitude-modulation channel; channel capacity; input-dependent additive Gaussian noise; multilevel flash memory; optimal coding rate.

I. Introduction

As the demand for non-volatile data storage increases, flash memories are gaining attention. The original flash memory used only two levels to store one bit in one memory cell. However, a modern mainstream flash memory is a multilevel flash memory (MLFM), which stores more than one bit in one memory cell to improve the storage density and reduce the bit cost of flash memories. In our previous work [1], we investigated the general MLFM with m-levels, where the number m of levels can be any integer not less than two. In practice, designers have presented some MLFMs, where the number m of levels are powers of two, such as the first MLFM product presented by Bauer *et al.* in [2], the 4-level MLFM in [3] and the Intel StrataFlash$^{\text{TM}}$ memory in [4], all three of which had four levels and stored two bits in one cell, the 8-level MLFMs in [5], [6] storing three bits in one cell, and the 16-level MLFMs in [7], [8] storing four bits in one cell.

It is obvious that, as the number of levels increases, the capability of the MLFM could be enhanced. However, due to the complexity of the configuration (including the programming/reading techniques and inter-cell interferences), it is complicated to model precisely the MLFM channel. Hence, research on the information-theoretic channel capacity is sporadic, such as [9], [10]. In particular, in [9], the simple upper and lower bounds in single letter formulas on the capacity were presented and computed numerically when the probability distribution of the channel inputs are assumed to be known. In [10], the MLFM was quantized to different discrete memoryless channels (DMCs) by introducing different reading numbers of reference voltages. By optimizing the reference voltages, the mutual information of DMC could be maximized, and then the achievable rate of the MLFM could be obtained.

Although the capability is enhanced, the reliability of the MLFM could be decreased because the margins between adjacent levels (voltages) in a cell are reduced as the number of levels increases and various interferences (for example, inter-cell interference) could arise. To guarantee the reliability, two approaches are usually investigated and applied in MLFMs. One approach is the on-chip error correcting technique [11]. Up to date, various error correcting codes (ECCs) used in the MLMC have been presented, such as the BCH codes [12], Reed-Solomon codes [13], [14], LDPC codes [10], [15], trellis coded modulation [12], [16], [17] and rank modulation [18], [19]. Other approaches are signal processing methods, for example, the data postcompensation method [9], the data pre-distortion method [9], and the coupling canceller method [20], which could tolerate the inter-cell interference in MLFMs.

To address the information-theoretic issues of the MLFM, we first need to solve a key problem, i.e., channel modeling. The simplest model is a constrained communication system, namely, an amplitude-limited *input-independent* additive white Gaussian noise (AWGN) channel, whose channel capacity and properties were investigated in [21], [22]. In [21], [22], Smith proved that the capacity of the amplitude-limited AWGN channel is achieved by a unique discrete random variable taking values on a finite alphabet. Based on the current techniques and configuration, there exist two universal phenomena for the MLFM. One is that the device degrades with age and the degradation varies from cell to cell as mentioned in [4], [23]. The other is the inter-cell interference as mentioned in [24]. In this paper, building upon our previous work [1], we are interested in only the former, while the latter was discussed in [9], [25]. The contribution of this work is two-fold. First, in Section II, we model the MLFM with m levels, also called m-level flash memory channel (m-LFMC) as an m-amplitude-modulation (m-AM) channel with input-dependent additive Gaussian noise (ID-AGN) whose standard deviation depends on the channel input (i.e., the voltage value of the level). The m-AM with ID-AGN channel can also be regarded as a constrained communication system [26], [27]. Second, we give the channel capacity and present a numerical method to evaluate it. Consequently, the optimal coding rate is obtained to guide the ECC code design.

Structure: The remainder of this paper is organized as follows. First, the m-LFMC is modeled as an m-AM channel with ID-AGN, as shown in Section II-A. Using channel output quantization, the channel can be considered as a discrete memoryless channel, as shown in Section II-B. Second, the channel capacities of the m-LFMC and the quantized channel are introduced in Sections III-A and III-B, respectively. The quantized capacity is an approximation of the capacity for the m-LFMC. Furthermore, the coding rate is defined in Section III-C. To evaluate the capacity, an alternating iterative algorithm is presented in Section IV, which delivers not only the optimal distribution of the channel input but also the optimal values of the channel input levels. Section V provides numerical results and discussions on the (quantized) capacities and optimal coding rates. We conclude this work in Section VI.

II. Channel Model

For an MLFM with m levels, each level has an intended *threshold voltage* [2]. By applying this voltage to the floating gate of a memory cell (transistor), the charge is maintained and then the data is stored in the cell. Affected by the configuration (including the programming/reading techniques) of the flash memory and device aging, the threshold voltage shift may vary from cell to cell. Hence, each level corresponds to a threshold voltage range [2]. In this paper, we focus on only the variation caused by device aging. For mathematical modeling, the variation of the threshold voltage is usually approximated by a Gaussian distribution and characterized by its probability density function (pdf). The following example illustrates the models of threshold voltage distributions for a 4-LFMC.

Example 1 (Threshold Voltage Distributions of a 4-LFMC): Consider a 4-LFMC. Let the intended voltages of the four levels be $x_0 = 0$, $x_1 = 3.25$, $x_2 = 4.55$ and $x_3 = 6.5$. Note that, throughout this paper, the voltage is measured in the unit of *volt*, which is omitted if no confusion arises. By default, the threshold voltage distribution model of the manufactured 4-LFMC is shown in Figure 1, where the noise at each level has the same variance and the pdf of the output for each level is depicted. As documented in [4], [23], the number of electrons of a cell decreases with time and some cells become defective as time elapses, which means that the cell has a long but finite lifetime and the degradation varies from cell to cell. Consequentially, the performance of the 4-LFMC gets gradually worse as the device ages. Suppose that, after three years, the threshold voltage distribution model of the 4-LFMC is shown in Figure 2, where every level experiences more noise than in Figure 1 and the first level x_0 is the most noisy level while the other three levels have almost the same noise. Again, suppose that, after five years, the threshold voltage distribution model of the 4-LFMC is shown in Figure 3, where every level has even more noise

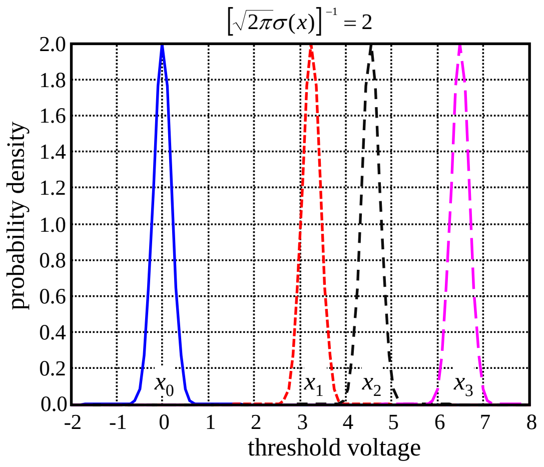


Figure 1. A threshold voltage distribution model for a 4-LFMC, in which the noise at each level has the same variance $\sigma(x) = \frac{1}{2\sqrt{2\pi}}$.

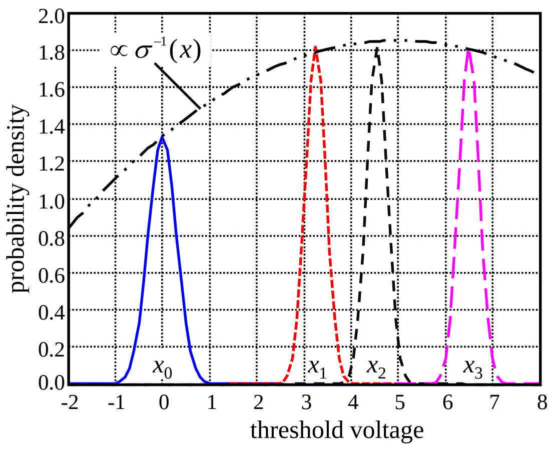


Figure 2. A threshold voltage distribution model for a 4-LFMC, in which the first level x_0 is the most noisy level while the other three levels have roughly the same noise.

than in Figure 2, while the first level x_0 and the last level x_3 are respectively the most noisy levels. This behavior can be easily modeled by a function $\sigma(x)$, which depends on the age of the device. As shown in Figures 2 and 3, the dash-dot-dot curve $\left[\sqrt{2\pi}\sigma(x)\right]^{-1}$ is (approximately) the envelope of the peaks of the level-output-pdfs. In Figure 1, the curve $\sigma(x)$ is assumed to be a constant, i.e., $\sigma(x) = \frac{1}{2\sqrt{2\pi}}$.

Models similar to Figures 2 and 3 for the 4-LFMC were introduced in [4], [12], [15]. In particular, in [4], [12], the model of the 2 bits/cell (i.e., 4-level) NOR flash memory showed that the first level x_0 had the highest noise variance and the last level x_3 had the second highest noise variance while the two middle levels had almost the same noise variances. In [15], the model of a 4-level NAND flash memory showed that, when no inter-cell interference occurred, the first level x_0 had the highest Gaussian noise and the other three levels had almost the same noises characterized by bounded Gaussian variables.

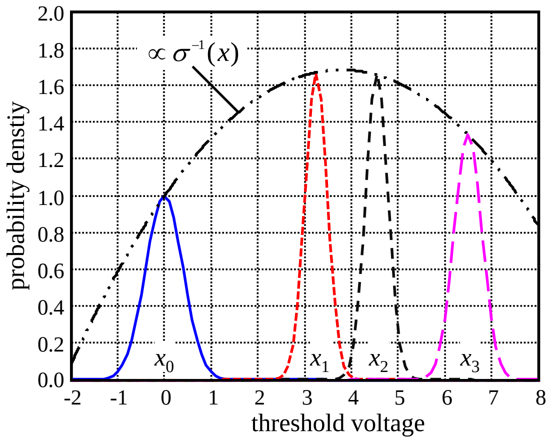


Figure 3. A threshold voltage distribution model for a 4-LFMC, in which the first level x_0 and the last level x_3 are respectively the most noisy levels while the two middle levels x_1 and x_2 have roughly the same noise.

A. *The ID-AGN m-AM Channel Model*

In this paper, an m-LFMC is modeled as an m-AM channel with ID-AGN. Specifically, it is characterized as follows.

1) Let X, Y and W denote the channel input, the channel output and the channel noise random variables, respectively. They have the relation:

$$Y = X + W. \tag{1}$$

2) The channel input X takes values from a finite alphabet $\mathcal{X}^{(m)} \triangleq \{x_0, x_1, \cdots, x_{m-1}\}$ under the constraint

$$a \le x_0 < x_1 < x_2 < \cdots < x_{m-2} < x_{m-1} \le b, \tag{2}$$

where a and b are the respective lowest and highest possible threshold voltages, and their difference is denoted by $V_m \triangleq b - a$. The finite alphabet $\mathcal{X}^{(m)}$ is called an m-AM signal set. Denote the collection of all such m-AM signal sets as $\mathscr{X}^{(m)}$, i.e., $\mathcal{X}^{(m)} \in \mathscr{X}^{(m)}$. In the following context, we also use the vector notation $\underline{x}$ to denote the m levels, i.e., $\underline{x} = (x_1, x_2, \cdots, x_{m-1})$.

3) The probability mass function (pmf) of X over $\mathcal{X}^{(m)}$ is denoted by $\underline{p} = (p_0, p_1, \cdots, p_{m-1})$ with $p_i = \Pr(X = x_i)$.

4) The noise W is an ID-AGN whose standard deviation depends on the realization of the channel input. That is, the noise W has mean zero and variance depending on the channel input $x \in \mathcal{X}^{(m)}$, i.e., $W \sim \mathcal{N}\big(0, \sigma^2(x)\big)$. In this paper, the function $\sigma(x)$ is assumed to be continuous and differentiable.

Therefore, the channel transition pdf, i.e., the channel law, is

$$f_{Y|X,\sigma(\cdot)}(y|x) = \frac{1}{\sqrt{2\pi}\sigma(x)} \exp\left\{-\frac{(y-x)^2}{2\sigma^2(x)}\right\}. \tag{3}$$

And the pdf of the channel output Y can be obtained as

$$f_{Y,\sigma(\cdot)}(y) = \sum_{i=0}^{m-1} p_i \, f_{Y|X,\sigma(\cdot)}(y|x_i). \tag{4}$$

Recall Example 1 of 4-AM channels with ID-AGN. At the time of manufacturing, the noise standard deviations for all levels are considered to be constant; see Figure 1. As the device ages, the noise standard deviations for different levels increase in different extents; see Figures 2 and 3. That is, the noise standard deviations for an aged device are level-dependent.

B. *Quantized Channel Model*

In practice, the channel output is often obtained by quantizing the real-valued channel output voltage Y. In this way, a discrete memoryless channel (DMC) of the m-LFMC is obtained when the channel inputs are known and fixed. Let $Q(\cdot)$ be a quantizer of real values and $\hat{Y}$ be the quantized channel output, i.e., $\hat{Y} = Q(Y)$. We assume that, using the quantizer $Q(\cdot)$, the set $\mathbb{R}$ is partitioned into a sequence of n intervals as

$$(r_0, r_1], (r_1, r_2], \cdots, (r_{n-2}, r_{n-1}], (r_{n-1}, r_n), \tag{5}$$

where r_0 and r_n may be finite or infinite. Each interval (r_j, r_{j+1}) is represented by a representation point y_j. Then the finite alphabet of quantized channel outputs is $\mathcal{Y} = \{y_0, y_1, \cdots, y_{n-1}\}$. The quantized DMC is characterized by the channel law (the channel transition probability) as

$$p_{\sigma(\cdot)}(y_j|x_i) = \int_{r_j}^{r_{j+1}} f_{Y|X,\sigma(\cdot)}(y|x_i)\, dy. \tag{6}$$

Consequently, the pmf of the quantized channel output $\hat{Y}$ can be obtained as

$$p_{\sigma(\cdot)}(y_j) = \sum_{i=0}^{m-1} p_i \, p_{\sigma(\cdot)}(y_j|x_i). \tag{7}$$

III. Channel Capacities and Optimum Coding Rates

From the previous section, i.e., Section II-A, we know that the m-LFMC is modeled as an m-AM channel with ID-AGN, parameterized by the m-AM signal set $\mathcal{X}^{(m)}$, the pmf $\underline{p} = (p_0, p_1, \cdots, p_{m-1})$ and the standard deviation function $\sigma(x)$. Therefore, to express the information-theoretic essentials of the m-LFMC, we introduce a new notation different slightly from the conventional one by inserting the subscript $\big(\mathcal{X}^{(m)}, \sigma(\cdot)\big)$ into the mutual information expression, i.e.,

$$I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y) \triangleq \sum_{i=0}^{m-1} \int_{-\infty}^{\infty} p_i \, f_{Y|X,\sigma(\cdot)}(y|x_i) \log\left(\frac{f_{Y|X,\sigma(\cdot)}(y|x_i)}{f_{Y,\sigma(\cdot)}(y)}\right) dy. \tag{8}$$

Similarly, the mutual information of the DMC is given as

$$I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;\hat{Y}) \triangleq \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} p_i \, p_{\sigma(\cdot)}(y_j|x_i) \log\left(\frac{p_{\sigma(\cdot)}(y_j|x_i)}{p_{\sigma(\cdot)}(y_j)}\right). \tag{9}$$

A. *The Channel Capacity of the m-LFMC*

Definition 1: The capacity of the m-LFMC with standard deviation function $\sigma(\cdot)$ is defined as

$$C_{m,\sigma(\cdot)} \triangleq \max_{\mathscr{X}^{(m)}, \{\underline{p}\}} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y), \tag{10}$$

where the maximum is taken over all possible m-AM signal sets $\mathcal{X}^{(m)} = \{x_0, x_1, \cdots, x_{m-1}\} \in \mathscr{X}^{(m)}$ satisfying

$$a \leq x_0 < x_1 < \cdots < x_{m-2} < x_{m-1} \leq b \tag{11}$$

and all possible pmfs $\underline{p} = (p_0, p_1, \cdots, p_{m-1})$ satisfying

$$p_i \geq 0, \text{ and } \sum_{i=0}^{m-1} p_i = 1. \tag{12}$$

Remark 1. Recall Smith's result that the capacity of the amplitude-limited AWGN channel is achieved by a unique discrete random variable taking values on a finite alphabet [21], [22]. The main two differences between the m-AM channel with ID-AGN and the amplitude-limited AWGN channel are: the noise in the former is input-dependent, while in the latter it is independent of inputs; and the number of inputs is fixed to be m in the former, while in the latter the optimal (capacity-achieving) number of inputs is obtained by optimization.

Remark 2. Comparing with Ungerboeck's results of average energy limited AWGN channel with amplitude modulation [28], there are three main differences. First, the m-AM channel with ID-AGN for an m-LFMC is not average energy limited but amplitude limited (in the interval $[a, b]$). Second, the m-AM signal set is not fixed but can be optimized in the evaluation of its capacity. Third, the input distribution is not uniform but can be optimized too.

One of the main objectives in capacity research is numerical evaluation. To this end, a comprehensive understanding is necessary and can provide a methodology of evaluation. The following proposition gives an insight into the capacity $C_{m,\sigma(\cdot)}$ of the m-LFMC.

Proposition 1: When $\underline{x}$ is given, the mutual information $I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)$ is concave with respect to (w.r.t.) $\underline{p}$; when $\underline{p}$ is given, the mutual information $I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)$ is continuous and differentiable w.r.t. $\underline{x}$.

Proof sketch: The mutual information is expressed as

$$\begin{aligned} & I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y) \\ & = h_{\mathcal{X}^{(m)},\sigma(\cdot)}(Y) - \sum_{i=0}^{m-1} p_i \log \sigma(x_i) - \frac{1}{2}\log(2\pi e) \end{aligned} \tag{13}$$

since the noise is input-dependent. When $\underline{x}$ is given, due to the linearity of $\sum p_i \log \sigma(x_i)$, we can prove that the mutual information is concave w.r.t. $\underline{p}$ by using the same method as in [29]. When $\underline{p}$ is given, the composition of elementary functions in (8) is continuous and differentiable w.r.t. $\underline{x}$ because $\sigma(x)$ is assumed to be continuous and differentiable. ■

B. Quantized Capacity

Denote by $\mathcal{Q}$ the collection of all possible quantizers, i.e., $\mathcal{Q} \triangleq \{Q(\cdot)\}$. Then an approximation of the channel capacity (10) is obtained as follows.

Definition 2: The *quantized capacity* of the m-LFMC with standard deviation function $\sigma(\cdot)$ is defined as

$$\hat{C}_{m,\sigma(\cdot)} \triangleq \max_{\mathscr{X}^{(m)},\{\underline{p}\},\mathcal{Q}} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;\hat{Y}), \tag{14}$$

where $I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;\hat{Y})$ is defined in (9), which is the mutual information between the channel input and the quantized channel output $\hat{Y}$ using the quantizer $Q(\cdot)$. The maximum in (14) is taken over all possible m-AM signal sets $\mathcal{X}^{(m)} = \{x_0, x_1, \cdots, x_{m-1}\} \in \mathscr{X}^{(m)}$ satisfying (11), all possible pmfs $\underline{p} = (p_0, p_1, \cdots, p_{m-1})$ satisfying (12) and all possible quantizers $Q(\cdot) \in \mathcal{Q}$.

If the channel input values $x_0, x_1, \ldots, x_{m-1}$ are known and the quantizer $Q(\cdot)$ is determined, then the quantized capacity in (14) is the capacity of a DMC (see the channel law in (6))

$$\hat{C}_{m,\sigma(\cdot)} \triangleq \max_{\{\underline{p}\}} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;\hat{Y}). \tag{15}$$

This capacity can be computed by the well-known Blahut-Arimoto algorithm [30], [31]. Through such capacities, the determination of quantization will be further discussed in Section V-C.

C. Equivalent Binary Code Rate of a Capacity-Achieving Code

As mentioned in the Introduction, ECCs are widely employed in MLFM products. So it is necessary to know the desired coding rate before we design a proper code. The capacity (10) provides an insight into how to pick the code rate of an equivalent binary code that achieves the capacity. Any binary code of a rate greater than the capacity achieving rate can not be used to guarantee the reliability of the MLFM channel; whereas, a binary code of the capacity-achieving rate can be constructed to achieve the capacity of the MLFM channel.

Definition 3: The *code rate* of an equivalent binary capacity-achieving code for the m-LFMC with standard deviation function $\sigma(\cdot)$ is defined as

$$R_m \triangleq \frac{C_{m,\sigma(\cdot)}}{\log_2 m}. \tag{16}$$

When the number of levels m is known (and fixed), then the code rate R_m serves as the upper limit of possible binary code rates that can guarantee reliable reception. If the number of levels m is undetermined, we have a chance to vary m, and thereby find a more appropriate code rate R_m that serves our design purposes. For instance, if $R_3 > R_4$ and $C_{3,\sigma(\cdot)} \approx C_{4,\sigma(\cdot)}$, we may want to use a binary code of the higher coding rate R_3, and thereby save on hardware complexity by using only $m = 3$ channel input levels as well as save on code complexity because binary codes of higher rates require fewer redundant bits. More discussion on this topic is provided in Section V-B.

IV. Evaluation of a Lower Bound on Capacity

To evaluate the capacity (10) of the m-LFMC, we turn to a two-step optimization problem

$$C_{m,\sigma(\cdot)} = \sup_{\underline{x}\in[a,b]^m} \sup_{\underline{p}\in[0,1]^m} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)$$
$$\text{subject to} \begin{cases} a \leq x_0 < x_1 < \cdots < x_{m-1} \leq b \\ p_i \geq 0, \ i \in \{0, 1, \cdots, m-1\} \\ \sum_{i=0}^{m-1} p_i = 1 \end{cases}. \tag{17}$$

To solve the two-step optimization problem (17), we turn to two sub-problems.

Sub-problem I.

$$\begin{aligned} C(\underline{x}) &= \max_{\underline{p}\in[0,1]^m} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y) \\ \text{subject to} & \begin{cases} p_i \geq 0,\ i \in \{0,1,\cdots,m-1\} \\ \sum_{i=0}^{m-1} p_i = 1 \end{cases} . \end{aligned} \quad (18)$$

When $\underline{x}$ is given, Sub-problem I is a conventional capacity problem for memoryless channel with finite inputs. Due to the concavity of the mutual information w.r.t. $\underline{p}$ shown in Proposition 1, the well-known algorithm, Blahut-Arimoto algorithm (BAA) [30]–[32] can be used to solve Sub-problem I.

Sub-problem II.

$$\begin{aligned} C(\underline{p}) &= \max_{\underline{x}\in[a,b]^m} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y) \\ &\text{subject to } a \leq x_0 < x_1 < \cdots < x_{m-1} \leq b \end{aligned} . \quad (19)$$

The Karush-Kuhn-Tucker (KKT) conditions [33] of the sub-problem are that there exists $\mathbf{v}^* = (\underline{x}^*, \lambda^*, \mu^*)$ such that

$$\begin{cases} \left.\frac{\partial I}{\partial x_0}\right|_{\mathbf{v}^*} &= -\lambda^*, & \\ \left.\frac{\partial I}{\partial x_{m-1}}\right|_{\mathbf{v}^*} &= \mu^*, & \\ \left.\frac{\partial I}{\partial x_i}\right|_{\mathbf{v}^*} &= 0, & i \in \{1,2,\cdots,m-2\} \\ x_0^* &\geq a, & \\ x_{m-1}^* &\leq b, & \\ x_{i-1}^* &< x_i^*, & i \in \{1,2,\cdots,m-1\} \\ \lambda^* &\geq 0, & \\ \mu^* &\geq 0, & \\ \lambda^*(x_0^* - a) &= 0, & \\ \mu^*(x_{m-1}^* - b) &= 0. & \end{cases} \quad (20)$$

Note that the solution of (20) may be sub-optimal (a local solution) since the concavity of the mutual information w.r.t. $\underline{x}$ is unknown. However, a better $\underline{x}$ with a greater mutual information can be obtained by solving (20). The method to find such a better $\underline{x}$ is shown as below.

For convenience, we denote the pdfs $f_{Y|X,\sigma(\cdot)}(y|x_i)$ in (3) and $f_{Y,\sigma(\cdot)}(y)$ in (4) and the mutual information $I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)$ in (8) as $f(y|x_i)$, $f(y)$ and $I(X;Y)$, respectively.

We compute partial derivatives of the mutual information $I(X;Y)$. To this end, we first compute the partial derivatives of the transition pdf $f(y|x_i)$ w.r.t x_i for all $i \in \{0,1,\cdots,m-1\}$ as

$$\frac{\partial f(y|x_i)}{\partial x_i} = \begin{cases} f(y|x_i)\left[-\frac{\sigma'(x_i)}{\sigma(x_i)} + \frac{y-x_i}{\sigma^2(x_i)} + \frac{(y-x_i)^2\sigma'(x_i)}{\sigma^3(x_i)}\right], & \text{if } i=j \\ 0, & \text{if } i\neq j \end{cases}, \quad (21)$$

where $\sigma'(x_i) \triangleq \frac{d\sigma(x)}{d\,x_i}$ denotes the derivative of $\sigma(x_i)$ w.r.t. x_i. Then, using (8) and (13), the partial derivatives of the mutual information w.r.t. x_i for all $i \in \{0,1,\cdots,m-1\}$ are obtained

$$\begin{aligned} \frac{\partial}{\partial x_i} I(X;Y) &= -\int_{-\infty}^{\infty} \frac{\partial}{\partial x_i}\left(f(y)\ln f(y)\right)dy - \frac{p_i\sigma'(x_i)}{\sigma(x_i)} \\ &= \left[-\frac{p_i\sigma'(x_i)}{\sigma^3(x_i)}\int_{-\infty}^{\infty} f(y|x_i)\ln f(y)dy\right]\cdot x_i^2 \\ &\quad + \left[\frac{2p_i\sigma'(x_i)}{\sigma^3(x_i)}\int_{-\infty}^{\infty} yf(y|x_i)\ln f(y)dy \right. \\ &\quad \left. + \frac{p_i}{\sigma^2(x_i)}\int_{-\infty}^{\infty} f(y|x_i)\ln f(y)dy\right]\cdot x_i \\ &\quad + \left[-\frac{p_i\sigma'(x_i)}{\sigma^3(x_i)}\int_{-\infty}^{\infty} y^2 f(y|x_i)\ln f(y)dy \right. \\ &\quad - \frac{p_i}{\sigma^2(x_i)}\int_{-\infty}^{\infty} yf(y|x_i)\ln f(y)dy \\ &\quad \left. + \frac{p_i\sigma'(x_i)}{\sigma(x_i)}\int_{-\infty}^{\infty} f(y|x_i)\ln f(y)dy - \frac{p_i\sigma'(x_i)}{\sigma(x_i)}\right] \\ &\triangleq A_i x_i^2 + B_i x_i + C_i. \end{aligned} \quad (22)$$

Solving the KKT conditions (20) is equivalent to finding quantities $(\underline{x}, \lambda, \mu)$ that satisfy the equalities

$$\begin{cases} A_0 x_0^2 + B_0 x_0 + (C_0 + \lambda) = 0 \\ \lambda(x_0 - a) = 0 \end{cases}, \quad (23a)$$

$$\begin{cases} A_{m-1} x_{m-1}^2 + B_{m-1} x_{m-1} + (C_{m-1} - \mu) = 0 \\ \mu(x_{m-1} - b) = 0 \end{cases}, \quad (23b)$$

$$A_i x_i^2 + B_i x_i + C_i = 0,\ i \in \{1,2,\cdots,m-2\}, \quad (23c)$$

and the inequalities

$$\begin{cases} \lambda \geq 0 \\ \mu \geq 0 \\ a \leq x_0 < x_1 < \cdots < x_{m-2} < x_{m-1} \leq b \end{cases}. \quad (24)$$

Note that all quantities A_i, B_i and C_i depend on the input vector $\underline{x}$ and the standard deviation function $\sigma(\cdot)$ when the pmf $\underline{p}$ is given. To find the solution to the KKT conditions (23) by an iterative method, we assume that quantities A_i, B_i and C_i are independent of x_i. Then Eqns. (23) have at most $9 \times 2^{m-2}$ solutions. Moreover, under the full constraints in (24), the number of solutions may be much less than $9 \times 2^{m-2}$ (This happens in our numerical computations). Based on (23) and (24), we employ an iterative method to find a solution. Suppose that the input vector $\underline{x}^{(k)}$ is known at the beginning of the k-th iteration. Then solve Eqns. (23). Pick those solutions that satisfy all constraints in (24), and from them choose the one with the highest information rate as the improved input vector $\underline{x}^{(k+1)}$.

Based on the two sub-problems, an alternating iterative scheme is presented to solve problem (17). At each iteration, the two-stage alternating strategy shown below is employed.

Stage 1. Fix $\underline{x}$. Use the BAA to obtain the optimal $\underline{p}^*$

$$\underline{p}^* = \arg\max_{\underline{p}} I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y). \quad (25)$$

Stage 2. Fix $\underline{p}$. Solve (20) to obtain a better $\underline{x}^*$ such that

$$I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)\big|_{\underline{x}^*} \geq I_{\mathcal{X}^{(m)},\sigma(\cdot)}(X;Y)\big|_{\underline{x}}. \quad (26)$$

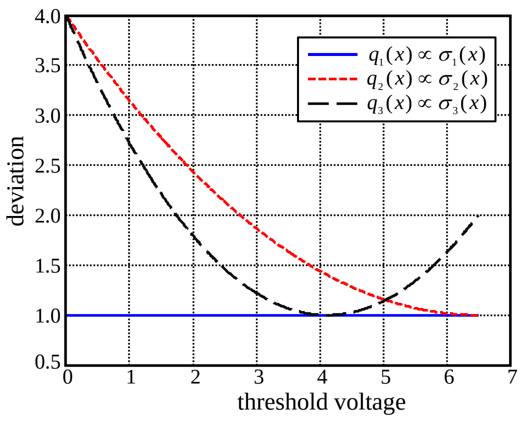


Figure 4. The standard deviation functions $\sigma_i(x)$ of the input-dependent Gaussian noise W: $\sigma_i(x) \propto q_i(x)$, where $i \in \{1, 2, 3\}$.

From the discussion of Sub-problem II, $\underline{x}^*$ may be a local solution. This sub-optimality also implies that a lower bound on the capacity $C_{m,\sigma(\cdot)}$ of the m-LFMC is evaluated.

V. Numerical Results and Discussions

In this section, we first numerically evaluate the capacities of different m-LFMCs using the alternating iterative scheme given in Section IV. We also interpret the results and put them in context with respect to prior works [21], [22]. Second, we estimate the optimal coding rate for the MLFM, which reveals a relationship between the capacity and the optimal number of levels. Third, the quantized capacities of the obtained DMCs using finite-level quantizations of the channel output are also numerically computed.

A. *Lower Bounds on the Capacity*

Let the lowest and highest threshold voltages be $a = 0$ and $b = 6.5$, respectively. Then the difference is $V_m = b - a = 6.5$. We introduce a new parameter $\sigma > 0$ that severs as the varying noise parameter in our computations. Let $q_i(x)$ where $i \in \{1, 2, 3\}$ be continuous and differentiable functions as shown in Figure 4. We consider three different standard deviation functions $\sigma(x)$, denoted as

$$\sigma_i(x) = q_i(x) \cdot \sigma, \text{ where } i \in \{1, 2, 3\}. \tag{27}$$

We allow the parameter σ to vary such that the *voltage-to-deviation ratio* (VDR) V_m/σ acts as an effective signal-to-noise ratio. We assume that the intended threshold voltage level x_0 (usually corresponding to the erased state) is 0.

We present results for $m \leq 5$, i.e., we consider multilevel flash memory channels with at most 5 levels. We consider three different m-LFMCs whose standard deviation functions are $\sigma_1(x)$, $\sigma_2(x)$ and $\sigma_3(x)$. The lower bounds on capacities of m-LFMCs with deviation functions $\sigma_1(x)$, $\sigma_2(x)$ and $\sigma_3(x)$ are shown in Figures 5, 6 and 7, respectively.

From Figure 5, we make the following observations.

1) When the VDR is less than 10.5 dB, i.e., $20\log_{10}(V_m/\sigma) \leq 10.5$ dB, 2-LFMC, 3-LFMC, 4-LFMC and 5-LFMC have the same rates.
2) When the VDR is less than 15 dB, 3-LFMC, 4-LFMC and 5-LFMC have the same rates.
3) When the VDR is less than 18 dB, 4-LFMC and 5-LFMC have the same rates.

Furthermore, we observe (not explicitly shown in the figure) that in the VDR regime between 10.5 dB and 15 dB, the optimized lower bound is achieved with $m^* = 3$ levels, even if, say, the constraint allows up to $m = 5$ levels. This implies that, for a fixed VDR, there is an optimal (minimal) number of levels m^* for a given MLFM channel. Increasing the number of levels m beyond m^* does not further increase the capacity (nor the computed lower bound).

Suppose that the number of levels is unknown. Then the MLFM channel is an amplitude-limited channel with ID-AGN, whose capacity is defined as

$$C_{\sigma(\cdot)} \triangleq \max_m C_{m,\sigma(\cdot)}, \tag{28}$$

where $\sigma(x)$ is the standard deviation function of the ID-AGN. The previous observations from Figure 5 imply that 2-LFMC, 3-LFMC and 4-LFMC can achieve the capacity $C_{\sigma_1(\cdot)}$ as defined in (28) in the cases of VDR $\leq$ 10.5 dB, 10.5 dB $<$ VDR $\leq$ 15 dB and 15 dB $<$ VDR $\leq$ 18 dB, respectively. In other words, at a given VDR less than 10.5 dB, a 2-LFMC is "optimal"; at a given VDR less than 15 dB, a 3-LFMC is "optimal"; at a given VDR less than 18 dB, a 4-LFMC is "optimal". Naturally, as the VDR increases, the optimal number of levels does not decrease. This is consistent with prior work [21], [22], which showed that for the amplitude-limited AWGN channel, the capacity is achieved by a discrete channel input distribution over a *finite* alphabet.

Similar conclusions hold for the other two channels with noise standard deviation functions $\sigma_2(x)$ and $\sigma_3(x)$. Namely, even if the constraint is set to be, say, $m = 5$, at low VDRs the optimal number of threshold levels m^* is less than 5. For example, as shown in Figure 7, the optimal number of levels is $m^* = 4$ in the VDR regime between 12 dB and 14.5 dB even when a 5-LFMC with noise standard deviation function $\sigma_3(x)$ is considered. In the case that VDR is equal to 14 dB, using the lower bound optimizing algorithm presented in Section IV, we obtain that the optimal number of levels is $m^* = 4$ with assignment $x_0^* = 0$, $x_1^* \approx 2.718$, $x_2^* \approx 4.212$ and $x_3^* = 6.5$ and pdf $p_0^* \approx 0.274$, $p_1^* \approx 0.171$, $p_2^* \approx 0.271$ and $p_3^* \approx 0.284$, shown in Figure 8. Again, this is consistent with the literature [21], [22] for the amplitude-limited AWGN channel, even though in m-LFMC the noise standard deviation $\sigma(x)$ is input-dependent.

B. *Optimal Binary Code Rate R^**

From the previous subsection, we know that for a given VDR, there is an optimal number of levels m^* that achieves that capacity $C_{\sigma(\cdot)}$ in (28). In other words, m^* is the minimal number of channel input levels required to achieve the capacity. Hence, the corresponding *optimal binary code rate* R^* is given by

$$R^* \triangleq \frac{C_{\sigma(\cdot)}}{\log_2 m^*}. \tag{29}$$

R^* is the rate of the equivalent binary capacity-achieving code that achieves the capacity using the smallest possible number

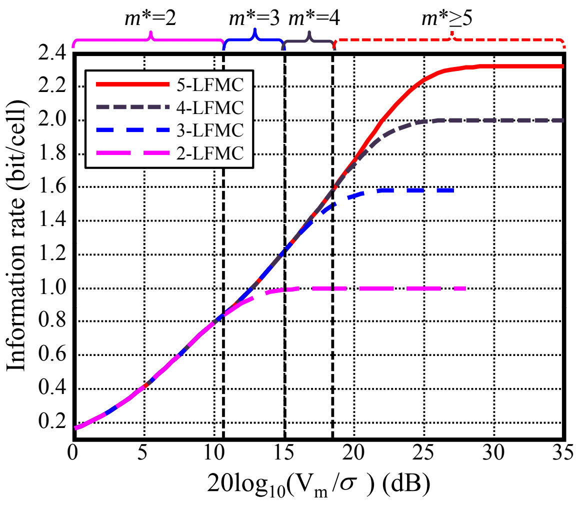


Figure 5. The information rates of m-LFMCs with $m \in \{2,3,4,5\}$ when the standard deviation function is $\sigma_1(x)$. The numbers m^* on the top of the figure indicate that 2-LFMC, 3-LFMC and 4-LFMC can achieve the (computed) maximum rates in the cases of VDR $\leq$ 10.5 dB, 10.5 dB $<$ VDR $\leq$ 15 dB and 15 dB $<$ VDR $\leq$ 18 dB, respectively.

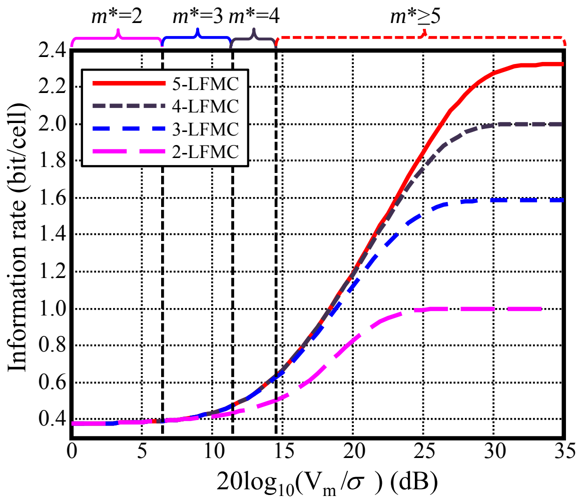


Figure 7. The achievable rates of m-LFMCs with $m \in \{2,3,4,5\}$ when the standard deviation function is $\sigma_3(x)$. The numbers m^* on the top of the figure indicate that 2-LFMC, 3-LFMC and 4-LFMC can achieve the (computed) maximum rates in the cases of VDR $\leq$ 6.5 dB, 6.5 dB $<$ VDR $\leq$ 11.5 dB and 11.5 dB $<$ VDR $\leq$ 14.5 dB, respectively.

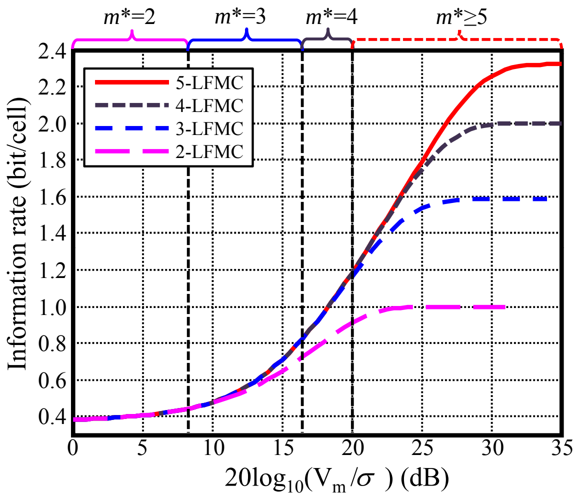


Figure 6. The information rates of m-LFMCs with $m \in \{2,3,4,5\}$ when the standard deviation function is $\sigma_2(x)$. The numbers m^* on the top of the figure indicate that 2-LFMC, 3-LFMC and 4-LFMC can achieve the (computed) maximum rates in the cases of VDR $\leq$ 8 dB, 8 dB $<$ VDR $\leq$ 16.5 dB and 16.5 dB $<$ VDR $\leq$ 20 dB, respectively.

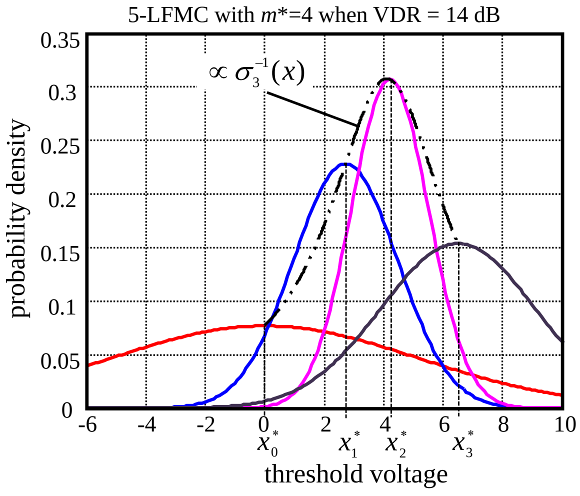


Figure 8. The pdfs of channel output distributions around the optimal threshold voltage levels when the m-LFMC with the standard deviation function $\sigma_3(x)$ and $m = 5$ is used at VDR = 14 dB. The optimal number of levels m^* is 4 with assignment $x_0^* = 0$, $x_1^* \approx 2.718$, $x_2^* \approx 4.212$ and $x_3^* = 6.5$ and pdf $p_0^* \approx 0.274$, $p_1^* \approx 0.171$, $p_2^* \approx 0.271$ and $p_3^* \approx 0.284$.

of levels m^*. Consequently, since m^* is the smallest number of levels that still guarantees the achievability of capacity, it follows that R^* is the highest possible rate of an equivalent binary code that can achieve the capacity $C_{\sigma(\cdot)}$. Hence, we refer to R^* as the *optimal rate*.

Figure 9 shows the optimal coding rate R^* for different optimized number of levels m^*, where $m^* \leq 8$, when the standard deviation function of the channel noise is $\sigma_1(x)$. Figure 10 shows the optimal coding rate R^* for different optimized number of levels m^*, where $m^* \leq 15$, when the standard deviation function of the channel noise is $\sigma_2(x)$. Figure 11 shows the optimal coding rate R^* for different optimized number of levels m^*, where $m^* \leq 15$, when the standard deviation function of the channel noise is $\sigma_3(x)$.

C. *Quantized Capacities*

In this subsection, we present the capacities of the quantized m-LFMC, where the values of the channel inputs are known and fixed and the standard deviation function of the noise is $\sigma_1(x)$. We explore an $(m * 2^k)$-level quantizer, in which the output is quantized into $(m * 2^k)$ intervals in a (non-uniform) way such that, around each level, there are 2^k equi-spaced intervals (which are indexed by k bits). An example of the quantization for the 4-LFMC with $x_0 = 0$, $x_1 = 3.25$, $x_2 = 4.55$ and $x_3 = 6.5$ is shown in Figure 12. Around each level, it is uniformly quantized into four intervals. Actually, it is a non-uniform quantization over $\mathbb{R}$. Then the channel becomes an DMC with four inputs and sixteen outputs. As mentioned in Section III-B, the quantized channel is a

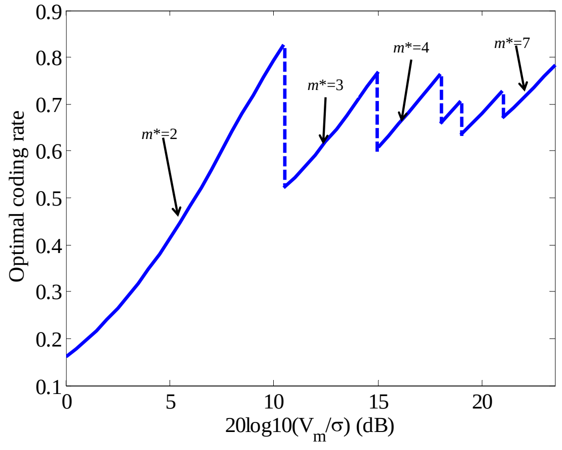


Figure 9. The optimal coding rates for m-LFMCs when the standard deviation function is $\sigma_1(x)$.

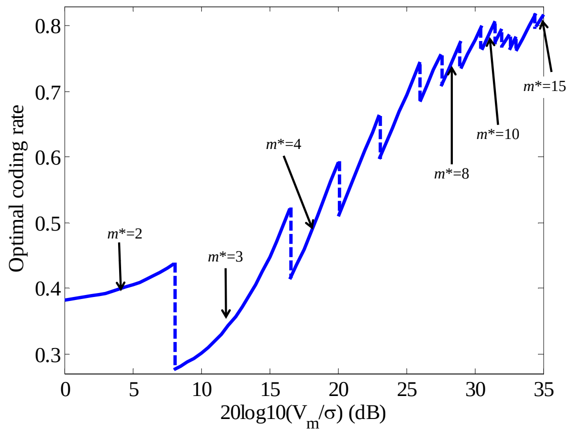


Figure 10. The optimal coding rates for m-LFMCs when the standard deviation function is $\sigma_2(x)$.

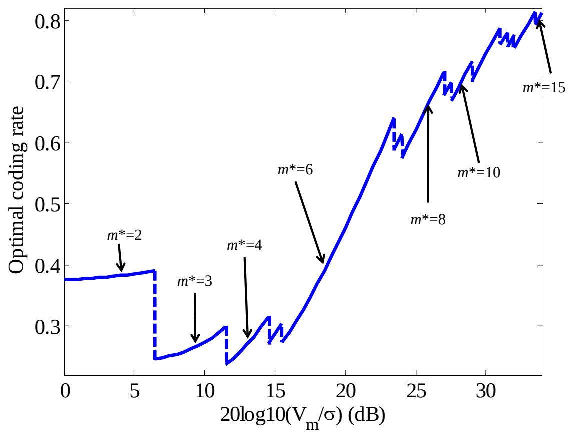


Figure 11. The optimal coding rates for m-LFMCs when the standard deviation function is $\sigma_3(x)$.

DMC whose capacity can be computed by the Blahut-Arimoto algorithm.

Suppose that six different $(m * 2^k)$-level quantizers, where $k = 0$, $k = 1$, $k = 2$, $k = 3$, $k = 4$ and $k = 5$ are used.

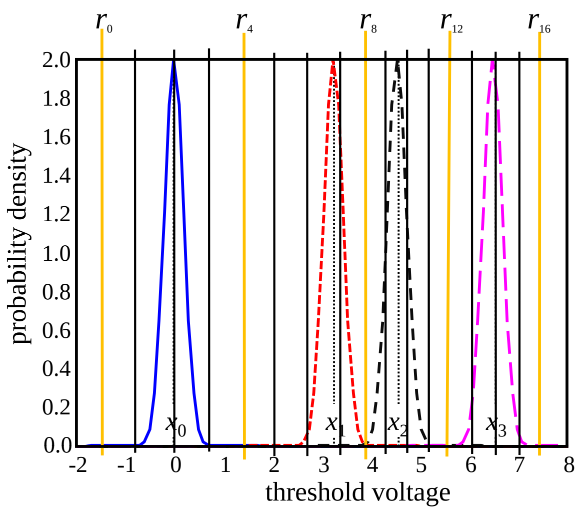


Figure 12. A (non-unform) 16-level quantization of the channel output for the 4-LFMC, where $x_0 = 0$, $x_1 = 3.25$, $x_2 = 4.55$ and $x_3 = 6.5$.

These quantizers are denoted by Q0, Q1, Q2, Q3, Q4 and Q5, respectively. Figures 13 and 14 show the capacities of different quantized DMCs for the 2-LFMC, where the channel inputs are fixed $\mathcal{X}^{(2)} = \{0, 6.5\}$. Figures 15 and 16 show the capacities of different quantized DMCs for the 4-LFMC, where the channel inputs are fixed $\mathcal{X}^{(4)} = \{0, 3.25, 4.55, 6.5\}$. Figures 17 and 18 show the capacities of different quantized DMCs for the 8-LFMC, where the channel inputs are fixed $\mathcal{X}^{(4)} = \{0, 1, 2, 3, 4, 5, 6, 6.5\}$. Also shown in these figures are the exact capacities without quantization. From these figures, we can see that,

1) as the number of quantization bits around each level (i.e., k) increases, the quantized capacity does not decrease;
2) in lower VDR regimes, a finite quantization induces some loss of capacity, as shown in Figures 13, 15, and 17;
3) in high VDR regimes, the quantized capacity of the 3-bit quantization around each level almost matches the exact capacity without quantization, as shown in Figures 14, 16 and 18.

Hence, for an m-LFMC with given channel inputs, the $m * 2^k$-level quantization (where k could be very small - at most 3) is good enough to practically approach the channel capacity.

VI. Conclusion

In this paper, the m-level flash memory was modeled as an m-AM channel with ID-AGN, in which the standard deviation of noise depends on the channel input. The capacity and the optimal coding rate of the m-LFMC were given. A simpler DMC was also derived by channel output quantization, which drove an approximation of the capacity for the m-LFMC. The determination of the capacity of the m-LFMC is an optimization problem, which can be transformed into two optimization sub-problems. One can be solved by the Blahut-Arimoto algorithm. The other can be solved by finding the solution to KKT conditions. Based on these, an alternating iterative algorithm was presented to evaluate a lower bound

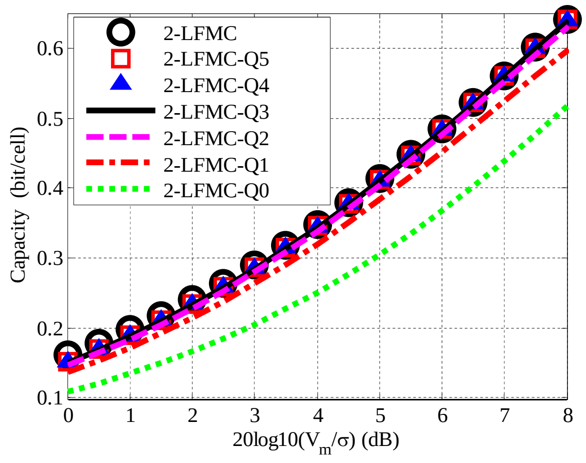


Figure 13. The capacities of different quantized DMCs for the 2-LFMC, where the channel inputs are fixed $\mathcal{X}^{(2)} = \{0, 6.5\}$.

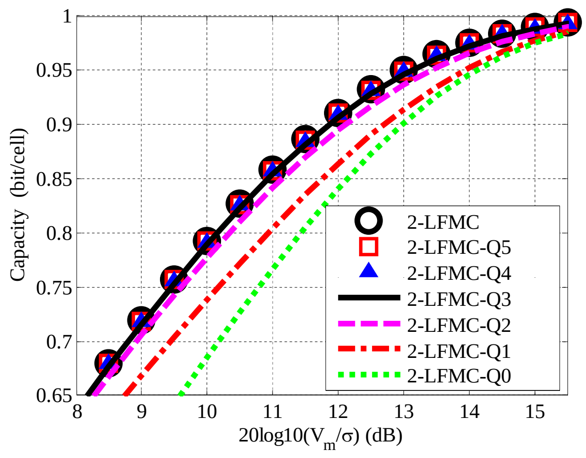


Figure 14. The capacities of different quantized DMCs for the 2-LFMC, where the channel inputs are fixed $\mathcal{X}^{(2)} = \{0, 6.5\}$.

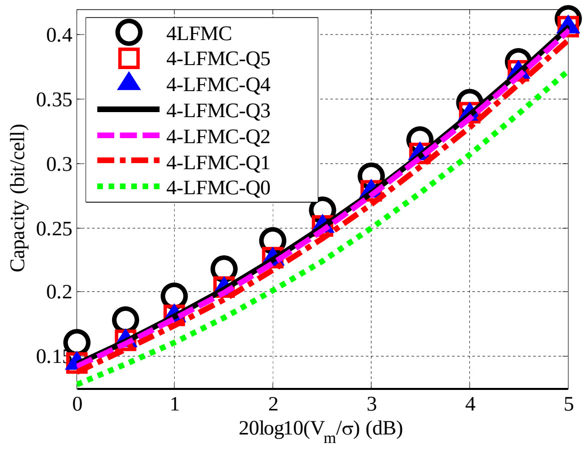


Figure 15. The capacities of different quantized DMCs for the 4-LFMC, where the channel inputs are fixed $\mathcal{X}^{(4)} = \{0, 3.25, 4.55, 6.5\}$.

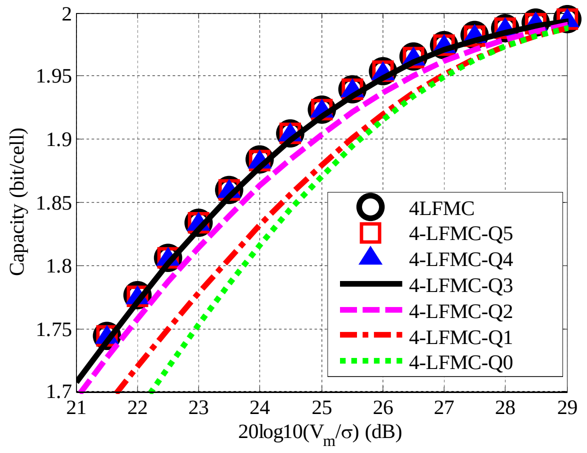


Figure 16. The capacities of different quantized DMCs for the 4-LFMC, where the channel inputs are fixed $\mathcal{X}^{(4)} = \{0, 3.25, 4.55, 6.5\}$.

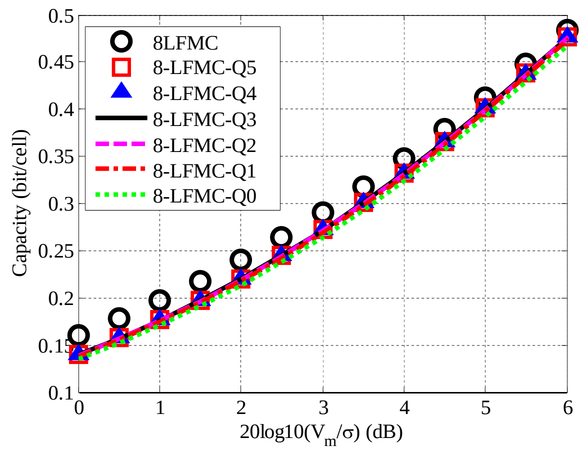


Figure 17. The capacities of different quantized DMCs for the 8-LFMC, where the channel inputs are fixed $\mathcal{X}^{(8)} = \{0, 1, 2, 3, 4, 5, 6, 6.5\}$.

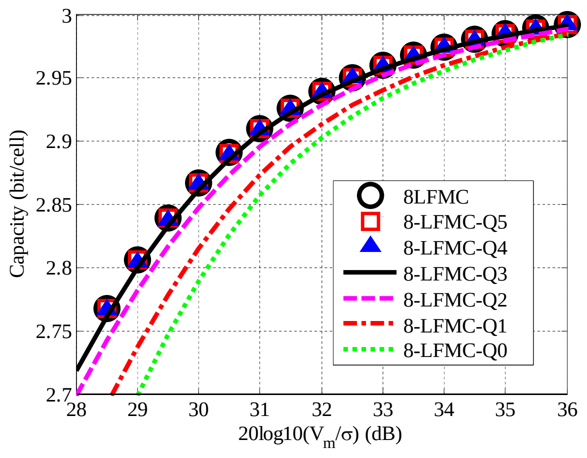


Figure 18. The capacities of different quantized DMCs for the 8-LFMC, where the channel inputs are fixed $\mathcal{X}^{(8)} = \{0, 1, 2, 3, 4, 5, 6, 6.5\}$.

on the capacity of the m-LFMC. This algorithm delivered not only the optimal distribution of channel inputs but also the optimal values of channel inputs. Numerical results showed that at any given VDR there exists an optimal (i.e., minimal) value m^* such that the capacity (or its lower bound) is achieved by an m^*-LFMC, and that increasing the number of levels m above m^* does not further increase the information rate for a fixed VDR. Numerical results also showed that if $(m * 2^k)$-

level quantizers with $k = 3$ are used at the channel output, the quantized capacity almost matches the capacity of the m-LFMC. Moreover, using the optimal coding rates as shown in the numerical results, we will design proper codes for the MLFM, which is one of our future works.

Acknowledgment

This work was supported by the collaborative NSF grants ECCS-1128705 and ECCS-1128148, and by the NSFC (No. 61172082).

References

[1] X. Huang, A. Kavcic, X. Ma, G. Dong, and T. Zhang, "Optimization of achievable information rates and number of levels in multilevel flash memories," in *ICN 2013: The Twelfth International Conference on Networks*, Seville, Spain, Jan. 27-Feb. 1 2013, pp. 125–131.

[2] M. Bauer, R. Alexis, and et al., "A multilevel-cell 32Mb flash memory," in *IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, Feb. 1995, pp. 132–133, 351.

[3] T.-S. Jung, Y.-J. Choi, and et al., "A 117-mm2 3.3-V only 128-Mb multilevel NAND flash memory for mass storage applications," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1575–1583, Nov. 1996.

[4] G. Atwood, A. Fazio, D. Mills, and B. Reaves, "Intel StrataFlash$^{\mathrm{TM}}$ memory technology overview," *Intel Technology Journal*, pp. 1–8, 4th Quarter 1997.

[5] Y. Li, S. Lee, and et al., "A 16Gb 3b/cell NAND flash memory in 56nm with 8MB/s write rate," in *IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, Feb. 2008, pp. 506–507.

[6] T. Futatsuyama, N. Fujita, and et al., "A 113mm2 32Gb 3b/cell NAND flash memory," in *IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, Feb. 2009, pp. 242–243.

[7] N. Shibata, H. Maejima, and et al., "A 70nm 16Gb 16-level-cell NAND flash memory," in *IEEE VLSI Circuits*, 2007, pp. 190–191.

[8] C. Trinh, N. Shibata, and et al., "A 5.6MB/s 64Gb 4b/cell NAND flash memory in 43nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, Feb. 2009, pp. 246–247, 247a.

[9] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst.–I: Reg. Papers*, vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[10] J. Wang, T. Courtade, H. Shankar, and R. D. Wesel, "Soft information for LDPC decoding in flash: mutual-information optimized quantization," in *Proc. IEEE GLOBECOM 2011*, Houston, Texas, USA, Dec. 2011.

[11] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 602–616, Apr. 2003.

[12] F. Sun, S. Devarajan, K. Rose, and T. Zhang, "Design of on-chip error correction systems for multilevel NOR and NAND flash memories," *IET Circuits Devices Syst.*, vol. 1, no. 3, pp. 241–249, 2007.

[13] J. Chen and P. H. Siegel, "Markov processes asymptotically achieve the capacity of finite-state intersymbol interference channels," *IEEE Trans. Inform. Theory*, vol. 54, no. 3, pp. 1295–1303, Mar. 2008.

[14] B. M. Kurkoshi, "The E8 lattice and error correction in multi-level flash memory," in *Proc. IEEE International Conference on Communications*, Kyoto, Japan, June 5-9 2011, pp. 1–5.

[15] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst.–I: Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[16] H. Lou and C. Sundberg, "Increasing storage capacity in multilevel memory cells by means of communications and signal processing techniques," *IEE Proc.-Circuits Devices Syst.*, vol. 147, no. 4, pp. 229–236, Aug. 2000.

[17] S. Soldà, D. Vogrig, A. Bevilacqua, A. Gerosa, and A. Neviani, "Analog decoding of trellis coded modulation for multi-level flash memories," in *Proc. the 2008 IEEE International Symposium on Circuits and Systems (ISCAS 2008)*, Seattle, U.S.A., May 18-21 2008, pp. 744–747.

[18] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. Inform. Theory*, vol. 55, no. 6, pp. 2659–2673, Jun. 2009.

[19] Z. Wang and J. Bruck, "Partial rank modulation for flash memories," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Austin, Texas, U.S.A., June 13-18 2010, pp. 864–868.

[20] D. Park and J. Lee, "Floating-gate coupling canceller for multi-level cell NAND flash," *IEEE Trans. Magn.*, vol. 47, no. 3, pp. 624–628, Mar. 2011.

[21] J. G. Smith, "On the information capacity of peak and average power constrained gaussian channels," Ph.D. dissertation, University of California, Berkeley, California, Dec. 1969.

[22] ——, "The information capacity of amplitude-and variance-constrained scalar Gaussian channels," *Information and Control*, vol. 18, pp. 203–219, 1971.

[23] Kingston, "Flash memory guide," Kingston, Tech. Rep., 2011.

[24] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Letters*, vol. 23, no. 5, pp. 264–266, May 2002.

[25] M. Asadi, X. Huang, A. Kavcic, and N. P. Santhanam, "Optimal detector for multilevel NAND flash memory channels with intercell interference," Nov. 2013, accepted by IEEE Journal on Seleted Areas in Communcation (J-SAC).

[26] S. Shamai, "Information theoretic aspects of constrained systems," in *MSRI Workshop on Information Theory*, Berkeley, California, U.S.A., Feb. 25 - Mar. 1 2002.

[27] ——, "Information theoretic aspects of constrained cell-sites cooperation," in *IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, Eilat, Israel, Nov. 17-20 2010, p. 000086.

[28] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. 28, no. 1, pp. 55–67, Jan. 1982.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons, Inc, 1991.

[30] R. E. Blahut, "Computation of channel capacity and rate distortion functions," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 4, pp. 460–473, Jul. 1972.

[31] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 1, pp. 14–20, Jan. 1972.

[32] A. Kavčić, "On the capacity of Markov sources over noisy channels," in *Proc. IEEE GLOBECOM 2001*, vol. 5, San Antonio, TX, USA, Nov. 25-29 2001, pp. 2997–3001.

[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

Enhanced Design Conditions for Decentralized State-Space Control of Systems with Relevant Interactions

Dušan Krokavec and Anna Filasová
Department of Cybernetics and Artificial Intelligence
Technical University of Košice, Faculty of Electrical Engineering and Informatics
Košice, Slovakia
dusan.krokavec@tuke.sk, anna.filasova@tuke.sk

***Abstract*—New points of view to the problems concerning the decentralized control of a class of large-scale systems with relevant subsystems interactions are presented in the paper. The problems are transformed into enhanced design conditions through slack matrices until global asymptotic stability of the complete system is pursued using Lyapunov approach. As results, a sufficient condition for the existence is formulated in terms of linear matrix inequalities while the impact of interconnection uncertainties is minimized using H$_\infty$ approach. The decentralized controllers proved to globally stabilize the system, both in noiseless and noisy conditions.**

Keywords-Large-scale systems; decentralized control; stabilizing conditions; linear matrix inequalities; H$_\infty$ robust control; control of multi-area power systems.

I. Introduction

Complex large-scale dynamic systems appear in many engineering fields and so, naturally, the control of large-scale systems has been studied by many researchers to provide comprehensive contributions on analytical and computational methods for feedback control design of such systems [2], [17]. Different decentralized control structures were proposed and different algorithms were derived depending on the local state control laws.

If the linear model of a large dynamic system is partitioned into interconnected subsystems, the interactions of the subsystem play significant role in global system stability and, if interactions contain uncertainties, expected performances cannot be attained if the control is designed only for the nominal models. The success of these methods can be improved if the system state are grouped so that subsystem interaction is minimized and the decentralized controllers are optimized with respect to interaction uncertainties. The first usable results for the existence of robust decentralized controllers mostly involve the conditions under which the matrix of interconnections in the considered large-scale system satisfies the prescribed matching condition [20], [23].

Recently, a number of efforts have been made to extend the application of robust control techniques using convex optimization, involving linear matrix inequalities (LMI). It is well known that LMI-based approaches [5] are powerful for a centralized control design, but, in the decentralized case, the control design task may not be oftentimes reducible to a feasibility problem because of existence of control law structural constraints.

To meet modern system requirements, controllers have to quarantine robustness over a wide range of system operating conditions and this further highlights the fact that robustness to interconnections and interaction uncertainties among subsystems is one of the major issues. Applying for power systems control, the most important terms are robustness and a decentralized control structure [15], [24]. The robustness issue arises to deal with uncertainties which mainly come from the varying network topology and the dynamic variation of the load. On the other hand, since a real-time information transfer among subsystems is unfeasible, decentralized controllers have to be exploited. To achieve less-conservative control gains design conditions, norm-bounded unknown uncertainties in subsystem interactions, or nonlinear bounds of interconnections, are included in LMI terms in the design condition formulation [9].

Focusing on the above problems, the paper is sequenced in eight sections and one appendix. Following the introduction in Section I, the second section places the results obtained within the context of existing requests. Section III briefly describes the problems concerning with control of the large-scale dynamical systems with relevant subsystem interactions. The preliminaries, mainly focused on the H_∞ based design approach as well as on the bounded real lemma forms, are presented in Section IV. Section V points out the stability analysis of the controlled system by use of a set of LMIs and Section VI states the newly proposed conditions for the state controller design. Section V illustrates the design task by numerical solutions and system stability analysis and Section VI draws some concluding remarks. Appendix is devoted to a model of the multi-area power systems, used in the illustrative example.

Throughout the paper, the notations is narrowly standard in such way that $\boldsymbol{x}^T$, $\boldsymbol{X}^T$ denotes the transpose of the vector $\boldsymbol{x}$ and matrix $\boldsymbol{X}$, respectively, $\boldsymbol{X} = \boldsymbol{X}^T > 0$, (≥ 0), means that $\boldsymbol{X}$ is a symmetric positive definite (semi-definite) matrix, *rank*$(\cdot)$ remits the rank of a matrix, the symbol $\boldsymbol{I}_n$ indicates the n-th order identity matrix, $I\!R$ denotes the set of real numbers, $I\!R^{n\times r}$ refers to the set of all $n\times r$ real matrices, $||\,.\,||$ entails the standard l_2-norm and $l_2\langle 0,+\infty)$ connotes the space of random signal over $\langle 0,+\infty)$.

II. The State of the Art

During the past decades, there has been significant but scattered activity in control of the systems with interactions. A necessary and sufficient condition for solvability, in the case of fixed interconnections, has been found, e.g., in [7], [8], [25], where a homotopic method was used to reduce the control design to a feasibility problem of a bilinear matrix inequality (BMI). Moreover, if the LMI method is adopted by using a single Lyapunov function [3], [19], it leads to very conservative results.

The paper reflects the problems concerning with the system robust stability for one class of disturbed large-scale systems, in the presence of interconnection uncertainties among subsystems. The used approach is concentrated on performance improvement of control systems and is a continuation of the earlier work started in [13], [18], especially motivated by the techniques presented in [4], and improved in [1] with respect to disturbance transfer function norm minimization, the system dynamics improvement and the decentralized control design simplification.

Comparing with the above mentioned articles, the merit of the results proposed in this paper relies on the conservatism reducing through slack matrices incorporation into enhanced design conditions. This represents issues which lead to a newly formulated set of LMIs, giving the sufficient conditions for design of the decentralized controllers, with closed-loop system matrix satisfying the Gershgorin circle theorem [11]. Results are illustrated using the load frequency control model of the multi-area power systems.

III. Problem Formulation

To formulate the control design task, it is assumed that the subsystems are given adequately to (A.10), (A.11), i.e., it is considered for $i = 1, 2, ..., p$ that

$$\dot{\boldsymbol{q}}_i(t) = \boldsymbol{A}_i \boldsymbol{q}_i(t) + \boldsymbol{b}_i u_i(t) + \sum_{l=1}^{p} \boldsymbol{G}_{il} \boldsymbol{q}_l(t) + \boldsymbol{f}_i d_i(t) \tag{1}$$

$$y_i(t) = \boldsymbol{c}_i^T \boldsymbol{q}_i(t) \tag{2}$$

where $\boldsymbol{q}_i(t) \in I\!R^{n_i}$ is the vector of the state variables of the i-th subsystem, $u_i(t), y_i(t) \in I\!R$ are input and output variables of the i-th subsystem, respectively, $\boldsymbol{A}_i, \boldsymbol{G}_{il} \in I\!R^{n_i \times n_i}$ are real matrices, $\boldsymbol{b}_i, \boldsymbol{c}_i, \boldsymbol{f}_i \in I\!R^{n_i}$ are real column vectors. The disturbance $d_i(t)$ is a non-anticipative precess, where $\{d(t) \in l_2(\langle 0, \infty); I\!R)$.

It is supposed that all states variables of a subsystem are measured or observed, all subsystems matrix of dynamics $\boldsymbol{A}_i$, $i = 1, 2, ..., p$ are of full rank, all pairs $(\boldsymbol{A}_i, \boldsymbol{b}_i)$ are controllable, and the i-th subsystem is controlled by the local state feedback control law

$$u_i(t) = \boldsymbol{k}_i^T \boldsymbol{q}_i(t) \tag{3}$$

where $\boldsymbol{k}_i \in I\!R^{n_i}$ is a constant gain vector.

It is supposed that the interconnections with the uncertainty terms in (1) can be, in general, written as

$$\boldsymbol{G}_i \boldsymbol{h}_i(\boldsymbol{q}(t)) = \sum_{l=1}^{p} \boldsymbol{G}_{il} \boldsymbol{q}_l(t) \tag{4}$$

where $\boldsymbol{h}_i(\boldsymbol{q}(t)) \in I\!R^{n_i}$ is a vector function, satisfying the inequality

$$\boldsymbol{h}_i^T(\boldsymbol{q}(t))\boldsymbol{h}_i(\boldsymbol{q}(t)) \le \varepsilon_i^{-1} \boldsymbol{q}^T(t) \boldsymbol{w}_i^T \boldsymbol{w}_i \boldsymbol{q}(t) \tag{5}$$

where $\varepsilon_i^{-1} > 0$, $\varepsilon_i \in I\!R$ is a scalar parameter, related to interconnection uncertainties, and $\boldsymbol{w}_i$ are constant vectors of appropriate dimensions.

Using the overall system state variable vector $\boldsymbol{q}(t)$, defined as follows

$$\boldsymbol{q}^T(t) = \begin{bmatrix} \boldsymbol{q}_1^T(t) & \boldsymbol{q}_2^T(t) & \cdots & \boldsymbol{q}_p^T(t) \end{bmatrix} \tag{6}$$

then (5) can be rewritten as

$$\begin{aligned} \sum_{l=1}^{p} \boldsymbol{h}_l^T(\boldsymbol{q}(t))\boldsymbol{h}_l(\boldsymbol{q}(t)) = \boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{h}(\boldsymbol{q}(t)) \le \\ \le \boldsymbol{q}^T(t) \left[\sum_{l=1}^{p} \varepsilon_l^{-1} \boldsymbol{w}_l^T \boldsymbol{w}_l \right] \boldsymbol{q}(t) \end{aligned} \tag{7}$$

The global system model with the subsystem interactions takes now the form

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{A}\boldsymbol{q}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{h}(\boldsymbol{q}(t)) + \boldsymbol{F}\boldsymbol{d}(t) \tag{8}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{q}(t) \tag{9}$$

where

$$\boldsymbol{y}^T(t) = [\ y_1(t) \quad y_2(t) \quad \cdots \quad y_p(t)\ ] \tag{10}$$

$$\boldsymbol{u}^T(t) = [\ u_1(t) \quad u_2(t) \quad \cdots \quad u_p(t)\ ] \tag{11}$$

$$\boldsymbol{d}^T(t) = [\ d_1(t) \quad d_2(t) \quad \cdots \quad \boldsymbol{d}_p(t)\ ] \tag{12}$$

$$\boldsymbol{A} = \text{diag} \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_p \end{bmatrix} \tag{13}$$

$$\boldsymbol{B} = \text{diag} \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_p \end{bmatrix} \tag{14}$$

$$\boldsymbol{G} = \text{diag} \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{G}_2 & \cdots & \boldsymbol{G}_p \end{bmatrix} \tag{15}$$

$$\boldsymbol{F} = \text{diag} \begin{bmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_1 & \cdots & \boldsymbol{f}_p \end{bmatrix} \tag{16}$$

$$\boldsymbol{C} = \text{diag} \begin{bmatrix} \boldsymbol{c}_1^T & \boldsymbol{c}_1^T & \cdots & \boldsymbol{c}_p^T \end{bmatrix} \tag{17}$$

where $\boldsymbol{q}(t) \in I\!R^n$, $\boldsymbol{u}(t), \boldsymbol{y}(t) \in I\!R^r$, $\boldsymbol{A}, \boldsymbol{G} \in I\!R^{n \times n}$, $\boldsymbol{B}, \boldsymbol{F} \in I\!R^{n \times r}$, $\boldsymbol{C} \in I\!R^{r \times n}$ and $\sum_{i=1}^{p} n_i = n$.

The goal is the parameter design of the control law for overall system

$$\boldsymbol{u}(t) = \boldsymbol{K}\boldsymbol{q}(t) \tag{18}$$

where $\boldsymbol{K} \in I\!R^{r \times n}$,

$$\boldsymbol{K} = \text{diag} \begin{bmatrix} \boldsymbol{k}_1^T & \boldsymbol{k}_2^T & \cdots & \boldsymbol{k}_p^T \end{bmatrix} \tag{19}$$

in such way that the controlled global large-scale system is stable.

IV. Preliminary Results

The main purpose of this section is to present the concept of system quadratic performance, based on the H$_\infty$ norm of e system transfer matrix. In that sense are proven and exploited the following results.

Proposition 1: If $\boldsymbol{M}$, $\boldsymbol{N}$ are matrices of appropriate dimensions, and $\boldsymbol{X}$ is a symmetric positive definite matrix of proper dimension, then

$$\boldsymbol{M}^T\boldsymbol{N}+\boldsymbol{N}^T\boldsymbol{M}\leq\boldsymbol{N}^T\boldsymbol{X}\boldsymbol{N}+\boldsymbol{M}^T\boldsymbol{X}^{-1}\boldsymbol{M} \tag{20}$$

Proof: [12] Since $\boldsymbol{X}=\boldsymbol{X}^T>0$, then

$$\big(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{M}-\boldsymbol{X}^{\frac{1}{2}}\boldsymbol{N}\big)^T\big(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{M}-\boldsymbol{X}^{\frac{1}{2}}\boldsymbol{N}\big)\geq 0 \tag{21}$$

$$\boldsymbol{M}^T\boldsymbol{X}^{-1}\boldsymbol{M}+\boldsymbol{N}^T\boldsymbol{X}\boldsymbol{N}-\boldsymbol{M}^T\boldsymbol{N}-\boldsymbol{N}^T\boldsymbol{M}\geq 0 \tag{22}$$

It is evident that (22) implies (20). This concludes the proof. ■

Definition 1: Let a linear multi input and multi output (MIMO) system is described in the the state-space form by the equation

$$\dot{\boldsymbol{q}}(t)=\boldsymbol{A}\boldsymbol{q}(t)+\boldsymbol{B}\boldsymbol{u}(t) \tag{23}$$

and the output relation

$$\boldsymbol{y}(t)=\boldsymbol{C}\boldsymbol{q}(t)+\boldsymbol{D}\boldsymbol{u}(t) \tag{24}$$

where $\boldsymbol{q}(t)\in I\!R^n$, $\boldsymbol{u}(t)\in I\!R^r$, and $\boldsymbol{y}(t)\in I\!R^m$ are vectors of the state, input and output variables, respectively, and $\boldsymbol{A}\in I\!R^{n\times n}$, $\boldsymbol{B}\in I\!R^{n\times r}$, $\boldsymbol{C}\in I\!R^{m\times n}$ and $\boldsymbol{D}\in I\!R^{m\times r}$ are real matrices. Then the transfer function matrix $\boldsymbol{G}(s)$ of the system (23), (24) is

$$\boldsymbol{G}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D} \tag{25}$$

Note, this definition is used only in this section.

The proof of announced lemmas in this section is based on the following result (see, e.g., the proof of Theorem 1 in [12]).

Proposition 2: (quadratic performance) If a stable system is described by the transfer function matrix (25) of the dimension $m\times r$, there exists such positive $\gamma\in I\!R$ that

$$\int_0^\infty(\boldsymbol{y}^T(v)\boldsymbol{y}(v)-\gamma\boldsymbol{u}^T(v)\boldsymbol{u}(v))\mathrm{d}v>0 \tag{26}$$

where $\boldsymbol{y}(t)\in I\!R^m$ is the vector of the system output variables, $\boldsymbol{u}(t)\in I\!R^r$ is the vector of the system input variables and γ is an upper bound of square of the H$_\infty$ norm of (25).

Proof: It is evident, from (25), that

$$\widetilde{\boldsymbol{y}}(s)=\boldsymbol{G}(s)\widetilde{\boldsymbol{u}}(s) \tag{27}$$

where $\widetilde{\boldsymbol{y}}(s)$, $\widetilde{\boldsymbol{u}}(s)$ stands for the Laplace transform of m dimensional output vector and r dimensional input vector, respectively. Then (27) implies

$$\|\widetilde{\boldsymbol{y}}(s)\|\leq\|\boldsymbol{G}(s)\|\|\widetilde{\boldsymbol{u}}(s)\| \tag{28}$$

with $\|\boldsymbol{G}(s)\|$ standing for the H_2 norm of the system transfer function matrix $\boldsymbol{G}(s)$. Since H_∞ norm property states

$$\frac{1}{\sqrt{m}}\|\boldsymbol{G}(s)\|_\infty\leq\|\boldsymbol{G}(s)\|\leq\sqrt{r}\|\boldsymbol{G}(s)\|_\infty \tag{29}$$

where $\|\boldsymbol{G}(s)\|_\infty$ is the H_∞ norm of the system transfer function matrix $\boldsymbol{G}(s)$, using the notation $\|\boldsymbol{G}(s)\|_\infty=\sqrt{\gamma}$, the inequality (29) can be rewritten as

$$0<\frac{1}{\sqrt{m}}\leq\frac{\|\widetilde{\boldsymbol{y}}(s)\|}{\sqrt{\gamma}\|\widetilde{\boldsymbol{u}}(s)\|}\leq\frac{\|\boldsymbol{G}(s)\|}{\sqrt{\gamma}}\leq\sqrt{r} \tag{30}$$

Thus, based on Parceval's theorem, (30) gives for $m\geq 1$

$$1\leq\frac{\|\widetilde{\boldsymbol{y}}(s)\|}{\sqrt{\gamma}\|\widetilde{\boldsymbol{u}}(s)\|}=\frac{\Big(\int\limits_0^\infty\boldsymbol{y}^T(v)\boldsymbol{y}(v)\mathrm{d}v\Big)^{\frac{1}{2}}}{\sqrt{\gamma}\Big(\int\limits_0^\infty\boldsymbol{u}^T(v)\boldsymbol{u}(v)\mathrm{d}v\Big)^{\frac{1}{2}}} \tag{31}$$

and, subsequently, it yields

$$\int_0^\infty\boldsymbol{y}^T(v)\boldsymbol{y}(v)\mathrm{d}v-\gamma\int_0^\infty\boldsymbol{u}^T(v)\boldsymbol{u}(v)\mathrm{d}v\geq 0 \tag{32}$$

that is the mapping from $\boldsymbol{u}(t)$ to $\boldsymbol{y}(t)$ is said to have the H_∞ norm less than $\sqrt{\gamma}$.

It is evident that (32) implies (26). This concludes the proof. ■

Before exploiting the principle of quadratic performance in the design of control (18), the following bounded real lemmas for the system (23), (24) are recalled.

Lemma 1: (bounded real lemma) System described by (23), (24) is asymptotically stable with the quadratic performance $\sqrt{\gamma}$ if there exist a symmetric positive definite matrix $\boldsymbol{P}\in I\!R^{n\times n}$ and a positive scalar $\gamma\in I\!R$ such that

$$\boldsymbol{P}=\boldsymbol{P}^T>0,\qquad\gamma>0 \tag{33}$$

$$\begin{bmatrix}\boldsymbol{P}\boldsymbol{A}+\boldsymbol{A}^T\boldsymbol{P} & \boldsymbol{P}\boldsymbol{B} & \boldsymbol{C}^T\\ * & -\gamma\boldsymbol{I}_r & \boldsymbol{D}^T\\ * & * & -\boldsymbol{I}_m\end{bmatrix}<0 \tag{34}$$

where $\boldsymbol{I}_r\in I\!R^{r\times r}$, $\boldsymbol{I}_m\in I\!R^{m\times m}$ are identity matrices of given dimensions, respectively.

Here, and hereafter, $*$ denotes the symmetric item in a symmetric matrix.

Proof: (compare [5], [18]) Since overall system (23), (24) is linear in $\boldsymbol{q}(t)$, using the Krasovskii theorem (see, e.g., [16]) and considering (32), the Lyapunov function $v(\boldsymbol{q}(t))$ can be considered as

$$\begin{aligned}v(\boldsymbol{q}(t))=\boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{q}(t)+\\ +\int\limits_0^t(\boldsymbol{y}^T(v)\boldsymbol{y}(v)-\gamma\boldsymbol{r}^T(v)\boldsymbol{u}(v))\mathrm{d}v>0\end{aligned} \tag{35}$$

where $\boldsymbol{P}=\boldsymbol{P}^T>0$, $\gamma>0$.

Thus, evaluating the derivative of $v(\boldsymbol{q}(t))$ with respect to t along a system trajectory, it yields

$$\begin{aligned}\dot{v}(\boldsymbol{q}(t))=\dot{\boldsymbol{q}}^T(t)\boldsymbol{P}\boldsymbol{q}(t)+\boldsymbol{q}^T(t)\boldsymbol{P}\dot{\boldsymbol{q}}(t)+\\ +\boldsymbol{y}^T(t)\boldsymbol{y}(t)-\gamma\boldsymbol{u}^T(t)\boldsymbol{u}(t)<0\end{aligned} \tag{36}$$

Therefore, the substitution of (23), (24) into (36) gives

$$\begin{aligned}\dot{v}(\boldsymbol{q}(t))=(\boldsymbol{A}\boldsymbol{q}(t)+\boldsymbol{B}\boldsymbol{u}(t))^T\boldsymbol{P}\boldsymbol{q}(t)+\\ +\boldsymbol{q}^T(t)\boldsymbol{P}(\boldsymbol{A}\boldsymbol{q}(t)+\boldsymbol{B}\boldsymbol{u}(t))-\gamma\boldsymbol{u}^T(t)\boldsymbol{u}(t)+\\ +(\boldsymbol{C}\boldsymbol{q}(t)+\boldsymbol{D}\boldsymbol{u}(t))^T(\boldsymbol{C}\boldsymbol{q}(t)+\boldsymbol{D}\boldsymbol{u}(t))<0\end{aligned} \tag{37}$$

and with the notation

$$\boldsymbol{q}_c^T(t) = \begin{bmatrix} \boldsymbol{q}^T(t) & \boldsymbol{u}^T(t) \end{bmatrix} \tag{38}$$

it is obtained

$$\dot{v}(\boldsymbol{q}(t)) = \boldsymbol{q}_c^T(t)\boldsymbol{P}_c\,\boldsymbol{q}_c(t) < 0 \tag{39}$$

where

$$\boldsymbol{P}_c = \begin{bmatrix} \boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A} & \boldsymbol{P}\boldsymbol{B} \\ * & -\gamma\boldsymbol{I}_r \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{C}^T\boldsymbol{D} \\ * & \boldsymbol{D}^T\boldsymbol{D} \end{bmatrix} < 0 \tag{40}$$

Since

$$\begin{bmatrix} \boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{C}^T\boldsymbol{D} \\ * & \boldsymbol{D}^T\boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}^T \\ \boldsymbol{D}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \geq 0 \tag{41}$$

applying Schur complement property to (41), then (40) implies (34). This concludes the proof. ■

From this results, the stability problem is reduced to find a Lyapunov matrix $\boldsymbol{P}$ and a parameter γ to stabilize the system and to guarantee the H_∞ norm attenuation between $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$.

Lemma 2: (enhanced bounded real lemma) System described by (23), (24) is asymptotically stable with the quadratic performance $\sqrt{\gamma}$ if for given positive $\delta \in I\!R$ there exist symmetric positive definite matrices $\boldsymbol{P}, \boldsymbol{S} \in I\!R^{n\times n}$, and a positive scalar $\gamma \in I\!R$ such that

$$\boldsymbol{P} = \boldsymbol{P}^T > 0, \quad \boldsymbol{S} = \boldsymbol{S}^T > 0, \quad \gamma > 0 \tag{42}$$

$$\begin{bmatrix} \boldsymbol{S}_1\boldsymbol{A}+\boldsymbol{A}^T\boldsymbol{S} & \boldsymbol{S}\boldsymbol{B} & \boldsymbol{P}-\boldsymbol{S}+\delta\boldsymbol{A}^T\boldsymbol{S} & \boldsymbol{C}^T \\ * & -\gamma\boldsymbol{I}_r & \delta\boldsymbol{B}^T\boldsymbol{S} & \boldsymbol{D}^T \\ * & * & -2\delta\boldsymbol{S} & \boldsymbol{0} \\ * & * & * & -\boldsymbol{I}_m \end{bmatrix} < 0 \tag{43}$$

Proof: (compare, e.g., [13]) Since (23) implies

$$\boldsymbol{A}\boldsymbol{q}(t) + \boldsymbol{B}\boldsymbol{u}(t) - \dot{\boldsymbol{q}}(t) = \boldsymbol{0} \tag{44}$$

then with positive definite symmetric matrices $\boldsymbol{S}_1, \boldsymbol{S}_2 \in I\!R^{n\times n}$ it yields

$$\left(\boldsymbol{q}^T(t)\boldsymbol{S}_1 + \dot{\boldsymbol{q}}^T(t)\boldsymbol{S}_2\right)\left(\boldsymbol{A}\boldsymbol{q}(t) + \boldsymbol{B}\boldsymbol{u}(t) - \dot{\boldsymbol{q}}(t)\right) = 0 \tag{45}$$

Thus, adding (45) as well as the transpose of (45) to (36) and substituting (24) in (36) results in

$$\begin{aligned} \dot{v}(\boldsymbol{q}(t)) = {} & \dot{\boldsymbol{q}}^T(t)\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{P}\dot{\boldsymbol{q}}(t) - \gamma\boldsymbol{u}^T(t)\boldsymbol{u}(t) + \\ & +(\boldsymbol{C}\boldsymbol{q}(t)+\boldsymbol{D}\boldsymbol{u}(t))^T(\boldsymbol{C}\boldsymbol{q}(t)+\boldsymbol{D}\boldsymbol{u}(t)) + \\ & +(\boldsymbol{A}\boldsymbol{q}(t)+\boldsymbol{B}\boldsymbol{u}(t)-\dot{\boldsymbol{q}}(t))^T(\boldsymbol{S}_1\boldsymbol{q}(t)+\boldsymbol{S}_2\dot{\boldsymbol{q}}(t)) + \\ & +(\boldsymbol{q}^T(t)\boldsymbol{S}_1+\dot{\boldsymbol{q}}^T(t)\boldsymbol{S}_2)(\boldsymbol{A}\boldsymbol{q}(t)-\boldsymbol{B}\boldsymbol{u}(t)-\dot{\boldsymbol{q}}(t)) < 0 \end{aligned} \tag{46}$$

Using the notation

$$\boldsymbol{q}_c^{\circ T}(t) = \begin{bmatrix} \boldsymbol{q}^T(t) & \boldsymbol{u}^T(t) & \dot{\boldsymbol{q}}^T(t) \end{bmatrix} \tag{47}$$

the inequality (46) can be written as

$$\dot{v}(\boldsymbol{q}(t)) = \boldsymbol{q}_c^{\circ T}(t)\boldsymbol{P}_c^\circ\boldsymbol{q}_c^\circ(t) < 0 \tag{48}$$

where

$$\boldsymbol{P}_c^\circ = \boldsymbol{P}_{c1}^\circ + \boldsymbol{P}_{c2}^\circ < 0 \tag{49}$$

$$\boldsymbol{P}_{c1}^\circ = \begin{bmatrix} \boldsymbol{S}_1\boldsymbol{A}+\boldsymbol{A}^T\boldsymbol{S}_1^T & \boldsymbol{S}_1\boldsymbol{B} & \boldsymbol{P}-\boldsymbol{S}_1+\boldsymbol{A}^T\boldsymbol{S}_2 \\ * & -\gamma\boldsymbol{I}_r & \boldsymbol{B}^T\boldsymbol{S}_2 \\ * & * & -2\boldsymbol{S}_2 \end{bmatrix} < 0 \tag{50}$$

$$\boldsymbol{P}_{c2}^\circ = \begin{bmatrix} \boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{C}^T\boldsymbol{D} & \boldsymbol{0} \\ * & \boldsymbol{D}^T\boldsymbol{D} & \boldsymbol{0} \\ * & * & \boldsymbol{0} \end{bmatrix} \tag{51}$$

Thus, setting

$$\boldsymbol{S}_1 = \boldsymbol{S}, \qquad \boldsymbol{S}_2 = \delta\boldsymbol{S} \tag{52}$$

and applying analogously Schur complement property to (51), then (49) implies (43). This concludes the proof. ■

The consequence of this Lemma is that of separating the Lyapunov matrix $\boldsymbol{P}$ from the system matric parameters, i.e., there is no product $\boldsymbol{P}\boldsymbol{A}$, $\boldsymbol{P}\boldsymbol{B}$ in the LMIs, which substantially reduces conservatism of solutions, especially if the system is linear with polytopic uncertainties.

Conversely, in the Lemma, a positive real scalar δ is involved in the LMIs as a prescribed constant design parameter. The procedure of adding scalar in LMIs has been widely explored in literature (see, e.g., [22]). Moreover, such a parameterization is often needed when converting BMI into linear ones.

V. State Control Design

Algorithms for solutions to (18), which includes the design in the sense of this paper, are the subject of this section.

Proposition 3: [1] The autonomous system from (8) is asymptotically stable with bounded quadratic performance if there exist symmetric positive definite matrices $\boldsymbol{P}_i \in I\!R^{n_i\times n_i}$ and positive scalars $\gamma_i, \lambda_i, \varepsilon_i \in I\!R$ such that for $i = 1, 2, \ldots, p$

$$\boldsymbol{P}_i = \boldsymbol{P}_i > 0,\ \gamma_i > 0,\ \lambda_i > 0,\ \varepsilon_i > 0 \tag{53}$$

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P}\boldsymbol{B} & \boldsymbol{P}\boldsymbol{F} & \boldsymbol{C}^T & \boldsymbol{P}\boldsymbol{G} & \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_p \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\varepsilon_1 & & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \\ * & * & * & * & * & * & & -\varepsilon_p \end{bmatrix} < 0 \tag{54}$$

where

$$\boldsymbol{\Phi} = \boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{P} \tag{55}$$

the matrices

$$\boldsymbol{P} = \text{diag}\begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{P}_2 & \cdots & \boldsymbol{P}_p \end{bmatrix} \tag{56}$$

$$\boldsymbol{\Gamma}_u = \text{diag}\begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_p \end{bmatrix} \tag{57}$$

$$\boldsymbol{\Gamma}_d = \text{diag}\begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_p \end{bmatrix} \tag{58}$$

are structured matrix variables, and all system matrix parameter structures are given in (13)–(17).

Proof: Defining Lyapunov function as follows

$$v(\boldsymbol{q}(t)) = \boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{q}(t) + \\ + \int_0^t \left(\boldsymbol{y}^T(v)\boldsymbol{y}(v) - \sum_{h=1}^{p} \left(\gamma_h u_h^2 + \lambda_h \boldsymbol{d}_h^2\right)\right) \mathrm{d}v \tag{59}$$

where $v(\boldsymbol{q}(t)) > 0$, $\boldsymbol{P} = \boldsymbol{P}^T > 0$ is given in (56), and $\gamma_h > 0$, $\lambda_h > 0$, $h = 1, 2, \dots p$, are introduced in (57).

Evaluating the derivative of $v(\boldsymbol{q}(t))$ with respect to t along the autonomous system trajectories, then with the notation (11), (12) it yields

$$\dot{v}(\boldsymbol{q}(t)) = \dot{\boldsymbol{q}}^T(t)\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{P}\dot{\boldsymbol{q}}(t) + \\ + \boldsymbol{y}^T(t)\boldsymbol{y}(t) - \begin{bmatrix} \boldsymbol{u}^T(t) & \boldsymbol{d}^T(t) \end{bmatrix} \boldsymbol{\Gamma} \begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{d}(t) \end{bmatrix} < 0 \tag{60}$$

where, with (57), (58)

$$\boldsymbol{\Gamma} = \operatorname{diag} \begin{bmatrix} \boldsymbol{\Gamma}_u & \boldsymbol{\Gamma}_d \end{bmatrix} \tag{61}$$

Therefore, the substitution of whole system model equations (8), (9) into (60) gives

$$\dot{v}(\boldsymbol{q}(t)) = \boldsymbol{q}^T(t)\boldsymbol{C}^T\boldsymbol{C}\boldsymbol{q}(t) + \\ + \left(\boldsymbol{A}\boldsymbol{q}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{h}(\boldsymbol{q}(t)) + \boldsymbol{F}\boldsymbol{d}(t)\right)^T \boldsymbol{P}\boldsymbol{q}(t) + \\ + \boldsymbol{q}^T(t)\boldsymbol{P}\left(\boldsymbol{A}\boldsymbol{q}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{h}(\boldsymbol{q}(t)) + \boldsymbol{F}\boldsymbol{d}(t)\right) - \\ - \begin{bmatrix} \boldsymbol{u}^T(t) & \boldsymbol{d}^T(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Gamma}_u & \\ & \boldsymbol{\Gamma}_d \end{bmatrix} \begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{d}(t) \end{bmatrix} < 0 \tag{62}$$

Subsequently, using the inequality (20) with $\boldsymbol{X} = \boldsymbol{I}$, it can be written

$$\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{G}\boldsymbol{h}(\boldsymbol{q}(t)) \leq \\ \leq \boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{h}(\boldsymbol{q}(t)) \tag{63}$$

and exploiting the inequality (5) then (63) gives

$$\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{G}\boldsymbol{h}(\boldsymbol{q}(t)) \leq \\ \leq \boldsymbol{q}^T(t)\boldsymbol{P}\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\sum_{l=1}^{p} \varepsilon_l^{-1}\boldsymbol{w}_l^T\boldsymbol{w}_h\boldsymbol{q}(t) \tag{64}$$

It is simple to see that introducing the notation

$$\boldsymbol{q}_c^{\bullet T}(t) = \begin{bmatrix} \boldsymbol{q}^T(t) & \boldsymbol{u}^T(t) & \boldsymbol{d}^T(t) \end{bmatrix} \tag{65}$$

negative (62) imply that

$$\dot{v}(\boldsymbol{q}(t)) \leq \boldsymbol{q}_c^{\bullet T}(t)\boldsymbol{P}_c^{\bullet}\boldsymbol{q}_c^{\bullet}(t) < 0 \tag{66}$$

where

$$\boldsymbol{P}_c^{\bullet} = \boldsymbol{P}_{c1}^{\bullet} + \boldsymbol{P}_{c2}^{\bullet} + \boldsymbol{P}_{c3}^{\bullet} < 0 \tag{67}$$

$$\boldsymbol{P}_{c1}^{\bullet} = \begin{bmatrix} \boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} & \boldsymbol{P}\boldsymbol{B} & \boldsymbol{P}\boldsymbol{F} \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d \end{bmatrix} \tag{68}$$

$$\boldsymbol{P}_{c2}^{\bullet} = \begin{bmatrix} \boldsymbol{C}^T\boldsymbol{C} + \boldsymbol{P}\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{P} & \boldsymbol{0} & \boldsymbol{0} \\ * & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \boldsymbol{0} \end{bmatrix} \tag{69}$$

$$\boldsymbol{P}_{c3}^{\bullet} = \sum_{l=1}^{p} \boldsymbol{P}_{c3l}^{\bullet} = \sum_{l=1}^{p} \begin{bmatrix} \boldsymbol{w}_l^T\varepsilon_l^{-1}\boldsymbol{w}_l & \boldsymbol{0} & \boldsymbol{0} \\ * & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \boldsymbol{0} \end{bmatrix} \tag{70}$$

Since it yields

$$\boldsymbol{P}_{c1}^{\bullet} = \begin{bmatrix} \boldsymbol{C}^T \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \geq 0 \tag{71}$$

$$\boldsymbol{P}_{c2}^{\bullet} = \begin{bmatrix} \boldsymbol{P}\boldsymbol{G} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}^T\boldsymbol{P} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \geq 0 \tag{72}$$

$$\boldsymbol{P}_{c3l}^{\bullet} = \begin{bmatrix} \boldsymbol{w}_l \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \varepsilon_l^{-1} \begin{bmatrix} \boldsymbol{w}_l^T & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \geq 0 \tag{73}$$

applying Schur complement property to (71)–(73) then (67) implies (54). This concludes the proof. ■

Inserting the closed-loop system matrix $\boldsymbol{A}_c \in I\!R^{n\times n}$ instead the system matrix $\boldsymbol{A}$ in (54), where

$$\boldsymbol{A}_c = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} \tag{74}$$

the bilinear matrix inequality is obtained. To transform this BMI into LMI, the next new theorem is proposed.

Theorem 1: The system (8), with output given by the relation (9), is stabilized with quadratic performance via the controller (18) if there exist symmetric positive definite matrices $\boldsymbol{X}_i \in I\!R^{n_i\times n_i}$, the matrices $\boldsymbol{Y}_i \in I\!R^{m_i\times n_i}$ and positive scalars $\gamma_i, \lambda_i, \varepsilon_i \in I\!R$ such that for all $i = 1, 2, \dots, p$

$$\boldsymbol{X}_i = \boldsymbol{X}_i > 0,\ \gamma_i > 0,\ \lambda_i > 0,\ \varepsilon_i > 0 \tag{75}$$

$$\begin{bmatrix} \widetilde{\boldsymbol{\Phi}} & \boldsymbol{B} & \boldsymbol{F} & \boldsymbol{X}\boldsymbol{C}^T & \boldsymbol{G} & \boldsymbol{X}\boldsymbol{w}_1 & \cdots & \boldsymbol{X}\boldsymbol{w}_p \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\varepsilon_1 & & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \\ * & * & * & * & * & * & & -\varepsilon_p \end{bmatrix} < 0 \tag{76}$$

where

$$\widetilde{\boldsymbol{\Phi}} = \boldsymbol{X}\boldsymbol{A}^T + \boldsymbol{A}\boldsymbol{X} - \boldsymbol{Y}^T\boldsymbol{B}^T - \boldsymbol{B}\boldsymbol{Y} \tag{77}$$

the matrices

$$\boldsymbol{X} = \operatorname{diag} \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \cdots & \boldsymbol{X}_p \end{bmatrix} \tag{78}$$

$$\boldsymbol{Y} = \operatorname{diag} \begin{bmatrix} \boldsymbol{Y}_1 & \boldsymbol{Y}_2 & \cdots & \boldsymbol{Y}_p \end{bmatrix} \tag{79}$$

and the matrices $\boldsymbol{\Gamma}_u$, $\boldsymbol{\Gamma}_d$ given in (57), (58), respectively, are structured matrix variables, and the system matrix parameter structures are specified in (13)–(17).

If the above conditions hold, the set of control gain matrices is given by

$$\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{X}^{-1} = \begin{bmatrix} \boldsymbol{k}_1^T & \boldsymbol{k}_2^T & \cdots & \boldsymbol{k}_p^T \end{bmatrix} \tag{80}$$

Proof: Inserting the closed-loop system matrix (74) into (55) gives

$$\boldsymbol{\Phi} = \boldsymbol{PA} - \boldsymbol{PBK} + \boldsymbol{A}^T\boldsymbol{P} - \boldsymbol{K}^T\boldsymbol{B}^T\boldsymbol{P} \tag{81}$$

Then, defining the transform matrix

$$\boldsymbol{T} = \text{diag}\begin{bmatrix} \boldsymbol{P}^{-1} & \boldsymbol{I}_r & \boldsymbol{I}_r & \boldsymbol{I}_r & \boldsymbol{I}_r & 1 & \cdots & 1 \end{bmatrix} \tag{82}$$

and pre-multiplying the left hand as well as the right hand side of (54) by (82), the next LMI is obtained

$$\begin{bmatrix} \widetilde{\boldsymbol{\Phi}} & \boldsymbol{B} & \boldsymbol{F} & \boldsymbol{P}^{-1}\boldsymbol{C}^T & \boldsymbol{G} & \boldsymbol{P}^{-1}\boldsymbol{w}_1 & \cdots & \boldsymbol{P}^{-1}\boldsymbol{w}_p \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\varepsilon_1 & & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \\ * & * & * & * & * & * & & -\varepsilon_p \end{bmatrix} < 0 \tag{83}$$

where

$$\widetilde{\boldsymbol{\Phi}} = \boldsymbol{P}^{-1}\boldsymbol{A}^T - \boldsymbol{P}^{-1}\boldsymbol{K}^T\boldsymbol{B}^T + \boldsymbol{AP}^{-1} - \boldsymbol{BKP}^{-1} \tag{84}$$

Introducing the LMI variables

$$\boldsymbol{P}^{-1} = \boldsymbol{X}, \quad \boldsymbol{KP}^{-1} = \boldsymbol{Y} \tag{85}$$

then (85) implies (79), and (83), (84) implies (76), (77), respectively. This concludes the proof. ■

Note, now the optimization problem in Theorem 1 can be solved using the standard LMI solvers.

VI. Enhanced Design Conditions

The idea of separating the Lyapunov matrix $\boldsymbol{P}$ from the system matric parameters is based on the method of Krasovskii and the theory of slack matrices [13]. In short, for the linear large-scale systems this method lies the next new stability conditions, formulated with respect to the subsystem interactions quadratic performances.

Theorem 2: The autonomous system from (8) is asymptotically stable with bounded quadratic performance if for given positive $\delta \in I\!\!R$ there exist symmetric positive definite matrices $\boldsymbol{P}_i, \boldsymbol{V}_i \in I\!\!R^{n_i \times n_i}$ and positive scalars $\gamma_i, \lambda_i, \epsilon_i \in I\!\!R$ such that for $i = 1, 2, \ldots, p$

$$\boldsymbol{P}_i = \boldsymbol{P}_i > 0,\ \boldsymbol{V}_i = \boldsymbol{V}_i > 0,\ \gamma_i > 0,\ \lambda_i > 0,\ \epsilon_i > 0 \tag{86}$$

$$\begin{bmatrix} \boldsymbol{\Lambda} & \boldsymbol{VB} & \boldsymbol{VF} & \boldsymbol{\Psi} & \boldsymbol{C}^T & \boldsymbol{VG} & \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_p \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \delta\boldsymbol{B}^T\boldsymbol{V} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \delta\boldsymbol{F}^T\boldsymbol{V} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & -\boldsymbol{\Pi} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ * & * & * & * & * & * & -\epsilon_1 & & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \\ * & * & * & * & * & * & * & & -\epsilon_p \end{bmatrix} < 0 \tag{87}$$

where

$$\boldsymbol{\Lambda} = \boldsymbol{VA} + \boldsymbol{A}^T\boldsymbol{V} \tag{88}$$

$$\boldsymbol{\Psi} = \boldsymbol{P} - \boldsymbol{V} + \delta\boldsymbol{A}^T\boldsymbol{V} \tag{89}$$

$$\boldsymbol{\Pi} = 2\delta\boldsymbol{V} - \delta^2\boldsymbol{VGG}^T\boldsymbol{V} \tag{90}$$

the matrix

$$\boldsymbol{V} = \text{diag}\begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 & \cdots & \boldsymbol{V}_p \end{bmatrix} \tag{91}$$

and the matrices $\boldsymbol{P}$, $\boldsymbol{\Gamma}_u$, $\boldsymbol{\Gamma}_d$ given in (56), (57), (58), respectively, are structured matrix variables, and the system matrix parameter structures are specified in (13)–(17).

Proof: Since (8) implies

$$\boldsymbol{Aq}(t) + \boldsymbol{Bu}(t) + \boldsymbol{Gh}(\boldsymbol{q}(t)) + \boldsymbol{Fd}(t) - \dot{\boldsymbol{q}}(t) = \boldsymbol{0} \tag{92}$$

then with positive definite symmetric block diagonal matrices $\boldsymbol{V}_1^\diamond$, $\boldsymbol{V}_2^\diamond \in I\!\!R^{n \times n}$ it yields

$$\left(\boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\right)\begin{pmatrix} \boldsymbol{Aq}(t) + \boldsymbol{Bu}(t) + \\ +\boldsymbol{Gh}(\boldsymbol{q}(t)) + \boldsymbol{Fd}(t) - \dot{\boldsymbol{q}}(t) \end{pmatrix} = 0 \tag{93}$$

Then, adding (93) and the transpose of (93) to (60) and subsequently substituting (9) in (60) results in

$$\begin{aligned} \dot{v}(\boldsymbol{q}(t)) &= \dot{\boldsymbol{q}}^T(t)\boldsymbol{Pq}(t) + \boldsymbol{q}^T(t)\boldsymbol{P}\dot{\boldsymbol{q}}(t) + \boldsymbol{q}^T(t)\boldsymbol{C}^T\boldsymbol{Cq}(t) + \\ &+ \begin{pmatrix} \boldsymbol{Aq}(t) + \boldsymbol{Bu}(t) + \\ +\boldsymbol{Gh}(\boldsymbol{q}(t)) + \boldsymbol{Fd}(t) - \dot{\boldsymbol{q}}(t) \end{pmatrix}^T (\boldsymbol{V}_1^\diamond\boldsymbol{q}(t) + \boldsymbol{V}_2^\diamond\dot{\boldsymbol{q}}(t)) + \\ &+ \left(\boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\right)\begin{pmatrix} \boldsymbol{Aq}(t) + \boldsymbol{Bu}(t) + \\ +\boldsymbol{Gh}(\boldsymbol{q}(t)) + \boldsymbol{Fd}(t) - \dot{\boldsymbol{q}}(t) \end{pmatrix} - \\ &- \begin{bmatrix} \boldsymbol{u}^T(t) & \boldsymbol{d}^T(t) \end{bmatrix}\begin{bmatrix} \boldsymbol{\Gamma}_u & \\ & \boldsymbol{\Gamma}_d \end{bmatrix}\begin{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{d}(t) \end{bmatrix} < 0 \end{aligned} \tag{94}$$

Subsequently, using the inequality (20) with $\boldsymbol{X} = \boldsymbol{I}$, it can be written

$$\begin{aligned} &\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{V}_1^\diamond\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond\boldsymbol{Gh}(\boldsymbol{q}(t)) \le \\ &\le \boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{h}(\boldsymbol{q}(t)) + \boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond\boldsymbol{GG}^T\boldsymbol{V}_1^\diamond\boldsymbol{q}(t) \end{aligned} \tag{95}$$

$$\begin{aligned} &\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{V}_2^\diamond\dot{\boldsymbol{q}}(t) + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\boldsymbol{Gh}(\boldsymbol{q}(t)) \le \\ &\le \boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{h}(\boldsymbol{q}(t)) + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\boldsymbol{GG}^T\boldsymbol{V}_2^\diamond\dot{\boldsymbol{q}}(t) \end{aligned} \tag{96}$$

and, exploiting (5), then (95), (96) give

$$\begin{aligned} &\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{V}_1^\diamond\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond\boldsymbol{Gh}(\boldsymbol{q}(t)) \le \\ &\le \boldsymbol{q}^T(t)\sum_{l=1}^{p}\varepsilon_l^{-1}\boldsymbol{w}_l^T\boldsymbol{w}_l\boldsymbol{q}(t) + \boldsymbol{q}^T(t)\boldsymbol{V}_1^\diamond\boldsymbol{GG}^T\boldsymbol{V}_1^\diamond\boldsymbol{q}(t) \end{aligned} \tag{97}$$

$$\begin{aligned} &\boldsymbol{h}^T(\boldsymbol{q}(t))\boldsymbol{G}^T\boldsymbol{V}_2^\diamond\dot{\boldsymbol{q}}(t) + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\boldsymbol{Gh}(\boldsymbol{q}(t)) \le \\ &\le \boldsymbol{q}^T(t)\sum_{l=1}^{p}\varepsilon_l^{-1}\boldsymbol{w}_l^T\boldsymbol{w}_l\boldsymbol{q}(t) + \dot{\boldsymbol{q}}^T(t)\boldsymbol{V}_2^\diamond\boldsymbol{GG}^T\boldsymbol{V}_2^\diamond\dot{\boldsymbol{q}}(t) \end{aligned} \tag{98}$$

respectively. Thus, introducing the notation

$$\boldsymbol{q}_c^{\diamond T}(t) = \begin{bmatrix} \boldsymbol{q}^T(t) & \boldsymbol{u}^T(t) & \boldsymbol{d}^T(t) & \dot{\boldsymbol{q}}^T(t) \end{bmatrix} \tag{99}$$

(94) can be rewritten as

$$\dot{v}(\boldsymbol{q}(t)) \le \boldsymbol{q}_c^{\diamond T}(t)\boldsymbol{P}_c^\diamond\boldsymbol{q}_c^\diamond(t) < 0 \tag{100}$$

where

$$\boldsymbol{P}_c^\diamond = \boldsymbol{P}_{c1}^\diamond + \boldsymbol{P}_{c2}^\diamond + \boldsymbol{P}_{c3}^\diamond < 0 \tag{101}$$

$$\boldsymbol{P}_{c1}^\diamond = \\ = \begin{bmatrix} \boldsymbol{V}_1^\diamond\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{V}_1^\diamond & \boldsymbol{V}_1^\diamond\boldsymbol{B} & \boldsymbol{V}_1^\diamond\boldsymbol{F} & \boldsymbol{P} - \boldsymbol{V}_1^\diamond + \boldsymbol{A}^T\boldsymbol{V}_2^\diamond \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \boldsymbol{B}^T\boldsymbol{V}_2^\diamond \\ * & * & -\boldsymbol{\Gamma}_d & \boldsymbol{F}^T\boldsymbol{V}_2^\diamond \\ * & * & * & -2\boldsymbol{V}_2^\diamond + \boldsymbol{V}_2^\diamond\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{V}_2^\diamond \end{bmatrix} \tag{102}$$

$$\boldsymbol{P}_{c2}^\diamond = \begin{bmatrix} \boldsymbol{C}^T\boldsymbol{C} + \boldsymbol{V}_1^\diamond\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{V}_1^\diamond & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ * & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \boldsymbol{0} & \boldsymbol{0} \\ * & * & * & \boldsymbol{0} \end{bmatrix} \tag{103}$$

$$\boldsymbol{P}_{c3}^\diamond = \sum_{l=1}^{p} \boldsymbol{P}_{c3l}^\diamond = \sum_{l=1}^{p} \begin{bmatrix} \boldsymbol{w}_l^T\epsilon_l^{-1}\boldsymbol{w}_l & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ * & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ * & * & \boldsymbol{0} & \boldsymbol{0} \\ * & * & * & \boldsymbol{0} \end{bmatrix} \tag{104}$$

$$\epsilon_l^{-1} = 2\varepsilon_l^{-1} \tag{105}$$

Analogously, setting

$$\boldsymbol{V}_1^\diamond = \boldsymbol{V}, \qquad \boldsymbol{V}_2^\diamond = \delta\boldsymbol{V} \tag{106}$$

and applying the Schur complement property to (103), (104), then (101) implies (87). This concludes the proof. ■

The following theorem presents the design of a continuous state feedback controller to decentralized stabilization of the system (9).

Theorem 3: The system (8), with output given by the relation (9), is stabilized with quadratic performance via the controller (18) if there exist symmetric positive definite matrices $\boldsymbol{T}_i, \boldsymbol{Z}_i \in I\!R^{n_i \times n_i}$, the matrices $\boldsymbol{W}_i \in I\!R^{m_i \times n_i}$ and positive scalars $\gamma_i, \lambda_i, \epsilon_i \in I\!R$ such that for all $i = 1, 2, \ldots, p$

$$\boldsymbol{T}_i = \boldsymbol{T}_i > 0,\ \boldsymbol{Z}_i = \boldsymbol{Z}_i > 0,\ \gamma_i > 0,\ \lambda_i > 0,\ \epsilon_i > 0 \tag{107}$$

$$\begin{bmatrix} \widetilde{\boldsymbol{\Lambda}} & \boldsymbol{B} & \boldsymbol{F} & \widetilde{\boldsymbol{\Psi}} & \boldsymbol{Z}\boldsymbol{C}^T & \boldsymbol{G} & \boldsymbol{Z}\boldsymbol{w}_1 \cdots & \boldsymbol{Z}\boldsymbol{w}_p \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \delta\boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \delta\boldsymbol{F}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & -\widetilde{\boldsymbol{\Pi}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & * & * & -\epsilon_1 & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ * & * & * & * & * & * & * & -\epsilon_p \end{bmatrix} < 0 \tag{108}$$

where

$$\widetilde{\boldsymbol{\Lambda}} = \boldsymbol{A}\boldsymbol{Z} - \boldsymbol{B}\boldsymbol{W} + \boldsymbol{Z}\boldsymbol{A}^T - \boldsymbol{W}^T\boldsymbol{B}^T \tag{109}$$

$$\widetilde{\boldsymbol{\Psi}} = \boldsymbol{T} - \boldsymbol{Z} + \delta(\boldsymbol{Z}\boldsymbol{A}^T - \boldsymbol{W}^T\boldsymbol{B}^T) \tag{110}$$

$$\widetilde{\boldsymbol{\Pi}} = 2\delta\boldsymbol{Z} - \delta^2\boldsymbol{G}\boldsymbol{G}^T \tag{111}$$

the matrices

$$\boldsymbol{T} = \text{diag}\left[\ \boldsymbol{T}_1 \quad \boldsymbol{T}_2 \quad \cdots \quad \boldsymbol{T}_p\ \right] \tag{112}$$

$$\boldsymbol{Z} = \text{diag}\left[\ \boldsymbol{Z}_1 \quad \boldsymbol{Z}_2 \quad \cdots \quad \boldsymbol{Z}_p\ \right] \tag{113}$$

$$\boldsymbol{W} = \text{diag}\left[\ \boldsymbol{W}_1 \quad \boldsymbol{W}_2 \quad \cdots \quad \boldsymbol{W}_p\ \right] \tag{114}$$

and the matrices $\boldsymbol{\Gamma}_u$, $\boldsymbol{\Gamma}_d$ given in (57), (58), respectively, are structured matrix variables, and the system matrix parameter structures are specified in (13)–(17).

If the above conditions hold, the set of control gain matrices is given by

$$\boldsymbol{K} = \boldsymbol{W}\boldsymbol{Z}^{-1} = \left[\ \boldsymbol{k}_1^T \quad \boldsymbol{k}_2^T \quad \cdots \quad \boldsymbol{k}_p^T\ \right] \tag{115}$$

Proof: Inserting the closed-loop system matrix (74) into (88), (89) gives

$$\boldsymbol{\Lambda} = \boldsymbol{V}\boldsymbol{A} - \boldsymbol{V}\boldsymbol{B}\boldsymbol{K} + \boldsymbol{A}^T\boldsymbol{V} - \boldsymbol{K}^T\boldsymbol{B}^T\boldsymbol{V} \tag{116}$$

$$\boldsymbol{\Psi} = \boldsymbol{P} - \boldsymbol{V} + \delta(\boldsymbol{A}^T\boldsymbol{V} - \boldsymbol{K}^T\boldsymbol{B}^T\boldsymbol{V}) \tag{117}$$

$$\boldsymbol{\Pi} = 2\delta\boldsymbol{V} - \delta^2\boldsymbol{V}\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{V} \tag{118}$$

Since the matrix $\boldsymbol{V}$ is supposed to be positive definite, it can be set up the next transform matrix

$$\boldsymbol{T} = \text{diag}\left[\ \boldsymbol{V}^{-1} \quad \boldsymbol{I}_r \quad \boldsymbol{I}_r \quad \boldsymbol{V}^{-1} \quad \boldsymbol{I}_r \quad \boldsymbol{I}_r \quad 1 \quad \cdots \quad 1\ \right] \tag{119}$$

Pre-multiplying the left hand and the right hand side of (87) by (119), then it yields

$$\begin{bmatrix} \widetilde{\boldsymbol{\Lambda}} & \boldsymbol{B} & \boldsymbol{F} & \widetilde{\boldsymbol{\Psi}} & \boldsymbol{V}^{-1}\boldsymbol{C}^T & \boldsymbol{G} & \boldsymbol{w}_1^\diamond \cdots & \boldsymbol{w}_p^\diamond \\ * & -\boldsymbol{\Gamma}_u & \boldsymbol{0} & \delta\boldsymbol{B}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & -\boldsymbol{\Gamma}_d & \delta\boldsymbol{F}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & -\widetilde{\boldsymbol{\Pi}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & * & -\boldsymbol{I}_r & \boldsymbol{0} \cdots & \boldsymbol{0} \\ * & * & * & * & * & * & -\epsilon_1 & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ * & * & * & * & * & * & * & -\epsilon_p \end{bmatrix} < 0 \tag{120}$$

where

$$\widetilde{\boldsymbol{\Lambda}} = \boldsymbol{A}\boldsymbol{V}^{-1} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{V}^{-1} + \boldsymbol{V}^{-1}\boldsymbol{A}^T - \boldsymbol{V}^{-1}\boldsymbol{K}^T\boldsymbol{B}^T \tag{121}$$

$$\widetilde{\boldsymbol{\Psi}} = \boldsymbol{V}^{-1}\boldsymbol{P}\boldsymbol{V}^{-1} - \boldsymbol{V}^{-1} + \delta(\boldsymbol{V}^{-1}\boldsymbol{A}^T - \boldsymbol{V}^{-1}\boldsymbol{K}^T\boldsymbol{B}^T) \tag{122}$$

$$\widetilde{\boldsymbol{\Pi}} = 2\delta\boldsymbol{V}^{-1} - \delta^2\boldsymbol{G}\boldsymbol{G}^T \tag{123}$$

$$\boldsymbol{w}_l^\diamond = \boldsymbol{V}^{-1}\boldsymbol{w}_l,\ l = 1, 2, \ldots\ p \tag{124}$$

Introducing the LMI variables

$$\boldsymbol{V}^{-1} = \boldsymbol{Z}, \quad \boldsymbol{K}\boldsymbol{V}^{-1} = \boldsymbol{W}, \quad \boldsymbol{V}^{-1}\boldsymbol{P}\boldsymbol{V}^{-1} = \boldsymbol{T} \tag{125}$$

then (120)-(123) implies (108)–(111), respectively. This concludes the proof. ■

In order to make this result applicable and operational, it is necessary to give the parameter δ and verify the obtained H$_\infty$ quadratic constraints. In addition, the size of this parameter can be used for tuning of the dynamics of the closed-loop system responses.

VII. Illustrative Example

To demonstrate the algorithm properties, the next subsystem parameters for $i = 1, 2, 3$ are used [18]

$$\boldsymbol{A}_i = \begin{bmatrix} -12.50 & 0.00 & -5.21 & 0.00 \\ 3.33 & -3.33 & 0.00 & 0.00 \\ 0.00 & 6.00 & -0.05 & -6.00 \\ 0.00 & 0.00 & 1.10 & 0.00 \end{bmatrix}, \quad \boldsymbol{b}_i = \begin{bmatrix} 12.5 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$\boldsymbol{c}_i^T = [0 \quad 0 \quad 1 \quad 0], \quad \boldsymbol{f}_i^T = [0 \quad 0 \quad -6 \quad 0]$$

and

$$\boldsymbol{G}_{ih} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -0.55 & 0 \end{bmatrix}, \quad \boldsymbol{g}_x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.55 \end{bmatrix}$$

$$\boldsymbol{G} = \text{diag}\,[\, \boldsymbol{g}_x \quad \boldsymbol{g}_x \quad \boldsymbol{g}_x \,]$$

$$\boldsymbol{w}_1^T = [\, 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \,]$$

$$\boldsymbol{w}_2^T = [\, 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \,]$$

$$\boldsymbol{w}_3^T = [\, 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \,]$$

Thus, solving (75), (76) with respect to the LMI matrix variables $\boldsymbol{X}_i, \boldsymbol{Y}_i, \gamma_i, \lambda_i, \varepsilon_i, i = 1, 2, 3$ using SeDuMi package for Matlab [21], the feedback gain matrix design problem was feasible with the results

$$\boldsymbol{X}_i = \begin{bmatrix} 14.7389 & 2.7960 & -1.9062 & 3.6230 \\ * & 5.0205 & -1.7327 & 3.4803 \\ * & * & 2.2244 & -0.6127 \\ * & * & * & 3.6596 \end{bmatrix}$$

$$\boldsymbol{Y}_i^T = [\, -12.9516 \quad 1.4183 \quad 0.1961 \quad -3.4268 \,]$$

$$\gamma_i = 16.9863, \ \lambda_i = 15.5306, \ \varepsilon_i = 11.5833$$

According to these matrix parameters, the local control laws are constructed with the gain vectors

$$\boldsymbol{k}_i^T = [\, -0.4870 \quad 3.5065 \quad 1.4243 \quad -3.5505 \,]$$

resulting in the stable decentralized state control, characteristic by the subsystem closed-loop matrix eigenvalues spectrum

$$\rho(\boldsymbol{A}_{ch}) = \{-0.2646 \ -3.2126 \ -3.1578 \pm 11.9004\text{i}\}$$

Solving (107), (108) for $\delta = 0.2$ with respect to the LMI matrix variables $\boldsymbol{T}_i, \boldsymbol{Z}_i, \boldsymbol{Y}_i, \gamma_i, \lambda_i, \epsilon_i, i = 1, 2, 3,$ the feasible solution gives out the common positive-definite matrix variables

$$\boldsymbol{T}_i = \begin{bmatrix} 48.1661 & -4.5647 & -2.5138 & 4.2776 \\ * & 5.3518 & -3.2610 & 1.7874 \\ * & * & 4.0874 & -1.1766 \\ * & * & * & 1.8135 \end{bmatrix}$$

$$\boldsymbol{Z}_i = \begin{bmatrix} 9.5165 & 0.7452 & -1.5398 & 1.1538 \\ * & 3.4365 & -2.4504 & 1.5058 \\ * & * & 3.1948 & -1.0502 \\ * & * & * & 1.4237 \end{bmatrix}$$

and the LMI parameters

$$\boldsymbol{W}_i^T = [\, -6.7570 \quad 2.1314 \quad 0.2832 \quad -0.5702 \,]$$

$$\gamma_i = 22.1502, \ \lambda_i = 37.2683, \ \epsilon_i = 10.4566$$

Note that by increasing the value of the tuning parameter δ the LMI solution may become infeasible.

The local state control with the obtained gain matrix

$$\boldsymbol{k}_i^T = [\, -0.5540 \quad 1.9349 \quad 0.8510 \quad -1.3508 \,]$$

insures the global system stability with decentralized closed-loop system matrix eigenvalues spectrum

$$\rho(\boldsymbol{A}_{ch}) = \{-1.0044 \ -3.4075 \ -2.2717 \pm 8.8048\text{i}\}$$

That, in the both cases, the resulting system time responses have small relative damping is not given by the attribute of the presented algorithms but implies from the characteristic properties of multiarea power systems.

VIII. Concluding Remarks

Decentralized robust control design for large scale systems with relevant subsystem interactions is formulated in the paper as an optimization problem and solved by LMIs. A conveyed characterization for the interaction bounds is presented and the sufficient condition for stabilizing decentralized robust control design are newly originated in the bounded real lemma as well as in the enhanced bounded real lemma structure, respectively. Since the theorems are newly derived, the proofs was necessary to include due to their original contributions.

The optimization principe, involving structured matrix variables in the linear matrix inequalities, takes into account the interaction bounds and the resulted decomposition gives enough flexibility to allow the inclusion of more general subsystem interaction structures and different output channels measurement gains.

The feasibility and effectiveness of enhanced bounded real lemma based control design are demonstrated using a multi-area model of the power system. It was shown that the global system can be locally asymptotically stabilizable by the decentralized state feedback laws, where application of the tuning parameter can improve dynamic system responses.

Acknowledgement

The work presented in this paper was supported by VEGA, the Grant Agency of the Ministry of Education and the Academy of Science of Slovak Republic under Grant No. 1/0256/11. This support is very gratefully acknowledged.

References

[1] D. Krokavec and A. Filasová, "Decentralized state-space control involving subsystem interactions," in *Proc. 8th Int. Conf. on Systems ICONS 2013*, Sevilla, Spain, pp. 13-18, 2013.

[2] A.C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Philadelphia: SIAM, PE, USA, 2005.

[3] L. Bakule, "Decentralized control. An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87-98, 2008.

[4] G.K. Bekefadu and I. Erlich, "Robust decentralized controller design for power systems using convex optimization involving LMIs," in *Prepr. 16th IFAC Word Congress*, Prag, Czech Republic, pp. 1743-1743, 2005.

[5] B. Boyd, L. El Ghaoui, E. Peron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Philadelphia: SIAM, PE, USA, 1994.

[6] W.C. Chan and Y.Y. Hsu, "Automatic generation control of interconnected power systems using variable-structure controllers," *IEE Proc. C*, vol. 128, no. 5, pp. 269-279, 1981.

[7] N. Chen, M. Ikeda, and W. Gu, "Design of robust H_∞ control for interconnected systems: A homotopy method," *Int. J. Control, Automation, and Systems*, vol. 3, no. 2, pp. 143-151, 2005.

[8] C. Cheng, B. Tang, Y. Cao, and Y. Sun, "Decentralized robust H_∞ control of uncertain large-scale systems with state-delays. LMIs approach," *Proc. American Control Conference*, Philadelphia, PE, USA, pp. 3111-3115, 1998.

[9] C. Dou, J. Yang, X. Li, T. Gui, and Y. Bi, "Decentralized coordinated control for large power system based on transient stability assessment," *Int. J. Electrical Power & Energy Systems*, vol. 46, no. 1, pp. 153-162, 2013.

[10] O.I. Elgert and C.E. Fosha, "Optimum megawatt-frequency control of multiarea electric energy system," *IEEE Trans. Power Apparatus and Systems*, vol. 89, no. 4, pp. 556-563, 1970.

[11] D.G. Feingold and R.S. Varga, "Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem," *Pacific J. Math.*, vol. 12, no. 4, 1241-1250, 1962.

[12] A, Filasová and D. Krokavec, "Pairwise control principle in large-scale systems," *Arch. Control Sciences*, vol. 21, no. 3, pp. 227-242, 2011.

[13] A, Filasová and D. Krokavec, "Partially decentralized design principle in large-scale system control," in *Recent Advances in Robust Control. Novel Approaches and Design Methods*, A. Mueller Ed., Rijeca: InTech, Croatia, pp. 361-388, 2011.

[14] C.E. Fosha and O.I. Elgert, "The megawatt-frequency control problem. A new approach via optimal control theory," *IEEE Trans. Power Apparatus and Systems*, vol. 89, no. 4, pp. 563-577, 1970.

[15] Y. Guo, D.J. Hill and Y. Wang, "Nonlinear decentralized control of large-scale power systems," *Automatica*, vol. 36, no. 9, pp. 1275-1289, 2000.

[16] W.M. Haddad and V. Chellaboina, *Nonlinear Dynamical Systems and Control. A Lyapunov-Based Approach*, Princeton: Princeton University Press, NJ, USA, 2008.

[17] M. Jamshidi *Large-Scale Systems: Modeling, Control and Fuzzy Logic*, Upper Saddle River: Prentice Hall, NJ, USA, 1997.

[18] D. Krokavec and A. Filasová, "Load frequency control involving subsystem interaction," in *Proc. 9th Int. Conf. Control of Power Systems CPS 2010*, Tatranske Matliare, Slovakia, pp. 1-8, 2010.

[19] J. Lunze, *Feedback Control of Large-Scale Systems*, Englewood Cliffs: Prentice Hall, NJ, USA, 1992.

[20] J. Mohammadpour and K.M. Grigoriadis, *Efficient Modeling and Control of Large-Scale Systems*, New York: Springer, NY, USA, 2010.

[21] D. Peaucelle, D. Henrion, Y. Labit, and K. Taitz, *User's Guide for SeDuMi Interface 1.04*, Toulouse: LAAS-CNRS, France, 2002.

[22] G. Pipeleers, B. Demeulenaerea, J. Sweversa, and L. Vandenbergheb, "Extended LMI characterizations for stability and performance of linear systems," *Systems & Control Letters*, vol.58, no. 7, pp. 510-518, 2009.

[23] D.D. Siljak, D.M. Stipanovic, and A.I. Zecevic, "Robust decentralized turbine/governor control using linear matrix inequalities," *IEEE Trans. Power Systems*, vol. 19, no. 3, pp, 1096-1103, 2004.

[24] Y. Wang, R. Zhou, and C. Wen, "Robust load-frequency controller design for power systems," *IEE Proc. C*, vol. 140, no. 1, pp. 11-16, 1993.

[25] G. Zhai, M. Ikeda and Y. Fujisaki, "Decentralized H_∞ controller design. A matrix inequality approach using a homotopy method," *Automatica*, vol. 37, no. 4, pp. 565-572, 2001.

Appendix

Considering a multi-area power system, the next analysis is based on the assumption that the electrical interconnections within each area of multi-area power system are so strong, at least in relation to ties with the neighboring areas that the whole area can be characterized only by a single frequency (see, e.g., [18] and the references therein). Therefore, it is supposed that the power equilibrium applied to the area i can be written as

$$\begin{aligned} T_{Pi}\tfrac{\mathrm{d}\Delta f_i(t)}{\mathrm{d}t} + \Delta f_i(t) + K_{Pk}\Delta P_{Tk}(t) = \\ = K_{Pi}\Delta P_{Gi}(t) - K_{Pi}\Delta P_{Di}(t) \end{aligned} \tag{A.1}$$

where T_{Pi} is the area model time constant (s), $\Delta f_i(t)$ is the area incremental frequency deviation (Hz), K_{Pi} is the area gain (Hz/pu MW), $\Delta P_{Ti}(t)$ is the incremental change of the total real power exported from the area (Hz/pu MW), $\Delta P_{Gi}(t)$ is the incremental change in generator output (Hz/pu MW), and $\Delta P_{Di}(t)$ is the unknown load disturbance (Hz/pu MW).

If the line losses are neglected, the individual line powers can be written in the form

$$\begin{aligned} P_{Ti}(t) = \tfrac{|V_i||V_\upsilon|}{X_{\upsilon i}P_{\upsilon i}}\sin(\delta_i(t) - \delta_\upsilon(t)) = \\ = P_{Ti\upsilon\,\max}\sin(\delta_i(t) - \delta_\upsilon(t)) \end{aligned} \tag{A.2}$$

$$V_i(t) = |V_i|\exp(j\delta_i(t)),\ V_\upsilon(t) = |V_\upsilon|\exp(j\delta_\upsilon(t)) \tag{A.3}$$

is the terminal bus voltage of the line, and $X_{\nu i}$ is its reactance.

When the phase angles deviate from their nominal values by the amounts $\Delta\delta_i$, $\Delta\delta_\nu$, respectively, the next approximation can be obtained [14]

$$\begin{aligned} \Delta P_{Ti}(t) = \\ = \tfrac{V_i||V_\upsilon|}{X_{\upsilon i}P_{\upsilon i}}\cos(\delta_{in}(t) - \delta_{\upsilon n}(t))(\Delta\delta_i(t) - \Delta\delta_\upsilon(t)) \end{aligned} \tag{A.4}$$

$$\begin{aligned} \Delta P_{Ti}(t) = \\ = 2\pi\tfrac{|V_i||V_\upsilon|}{X_{\upsilon i}P_{\upsilon i}}\cos(\delta_{in}(t) - \delta_{\upsilon n}(t))\left\{\begin{matrix}\int_0^t \Delta f_i(r)\mathrm{d}r - \\ -\int_0^t \Delta f_\upsilon(r)\mathrm{d}r\end{matrix}\right\} \end{aligned} \tag{A.5}$$

respectively. Related to the area frequency changes, the time derivative of the individual line powers is

$$\frac{\mathrm{d}\Delta P_{Ti\upsilon}(t)}{\mathrm{d}t} = S_{i\upsilon}(\Delta f_i(t) - \Delta f_\upsilon(t)) \tag{A.6}$$

$$\frac{\mathrm{d}\Delta P_{Ti}(t)}{\mathrm{d}t} = \sum_{i\neq l} S_{il}(\Delta f_i(t) - \Delta f_l(t)) \tag{A.7}$$

respectively, where S_{il} is the synchronizing coefficient (electrical stiffness of the tie line).

The incremental generated power of the area i for small system variable changes around the nominal settings can be represented by the equations

$$T_{Ti}\frac{\mathrm{d}\Delta P_{Gi}(t)}{\mathrm{d}t} + \Delta P_{Gi}(t) = \Delta x_{Hi}(t) \tag{A.8}$$

$$T_{Hi}\frac{\mathrm{d}\Delta x_{Hi}(t)}{\mathrm{d}t} + \Delta x_{Hi}(t) = \Delta P_{Ci}(t) - \frac{1}{R_i}\Delta f_i(t) \tag{A.9}$$

where T_{Ti} is the turbine time constant (s), T_{Hi} is the governor time constant (s) (generator response is instantaneous), R_i is a measure of static speed droop (Hz/pu MW), $\Delta P_{Ci}(t)$ is the incremental change of the command signal to the speed changer (control input), and $\Delta x_{Hi}(t)$ is the incremental change in the governor value position (pu MW), all with respect to the area i.

From the analysis made above the given formulation shows that the functioning of the multiarea power system is roughly a process with relevant interactions. The compact notation of (A.1), (A.7), (A.8), and (A.9) in the state-space form so leads to the equations [18]

$$\dot{\boldsymbol{q}}_i(t)=\boldsymbol{A}_i\boldsymbol{q}_i(t)+\boldsymbol{b}_i u_i(t)+\sum_{l=1}^{p}\boldsymbol{G}_{li}\boldsymbol{q}_i(t)+\boldsymbol{f}_i d_i(t) \tag{A.10}$$

$$y_i(t)=\boldsymbol{c}_i^T\boldsymbol{q}_i(t) \tag{A.11}$$

where

$$\boldsymbol{q}_i(t)=[\,\Delta x_{Hi}(t)\ \ \Delta P_{Gi}(t)\ \ \Delta f_i(t)\ \ \Delta P_{Ti}(t)\,]^T \tag{A.12}$$

$$u_i(t)=\Delta P_{Ci}(t) \tag{A.13}$$

$$d_i(t)=\Delta P_{Di}(t) \tag{A.14}$$

$$\boldsymbol{A}_i=\begin{bmatrix} -\frac{1}{T_{Hi}} & 0 & -\frac{1}{R_iT_{Hi}} & 0 \\ \frac{1}{T_{Ti}} & -\frac{1}{T_{Ti}} & 0 & 0 \\ 0 & \frac{K_{Pi}}{T_{Pi}} & -\frac{1}{T_{Pi}} & -\frac{K_{Pi}}{T_{Pi}} \\ 0 & 0 & \sum\limits_{l\neq i} S_{il} & 0 \end{bmatrix} \tag{A.15}$$

$$\boldsymbol{G}_{li}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -S_{li} & 0 \end{bmatrix},\quad \boldsymbol{b}_i=\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{T_{Hi}} \end{bmatrix} \tag{A.16}$$

$$\boldsymbol{f}_i=\begin{bmatrix} 0 \\ 0 \\ -\frac{K_{Pi}}{T_{Pi}} \\ 0 \end{bmatrix},\quad \boldsymbol{c}_i=\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{A.17}$$

More details, or multiarea model structure modifications, can be found, e.g., in [6], [10].

Under above given model parameters, the stability of the overall system can be studied by the stability properties of all subsystems, and by global features of all subsystems interactions.

Rapid Aerial Mapping with Multiple Heterogeneous Unmanned Vehicles

Eduard Santamaria, Florian Segor, Igor Tchouchenkov, and Rainer Schönbein
IAS – Interoperabilität und Assistenzsysteme
Fraunhofer IOSB
Karlsruhe, Germany
{eduard.santamaria, florian.segor, igor.tchouchenkov, rainer.schoenbein}@iosb.fraunhofer.de

Abstract—**One focus of research at Frauhofer IOSB is the utilization of unmanned aerial vehicles for data acquisition. Past efforts have lead to the development of a hardware and software system able to rapidly generate a complete and up-to-date aerial image by combining several single high resolution pictures taken by multiple unmanned aerial vehicles. However, the path planning component of the system was not designed to support no-fly zones inside the area of interest. Besides, the system assumed that all vehicles would have equal flight range and the same sensor footprint. In this paper, we address these limitations and present a new complete coverage path planning algorithm with support for no-fly zones inside the area of interest. The proposed method is suitable for non-convex areas, possibly with holes, to be covered by one or more maneuverable systems such as multi-rotor aircraft. Range and sensor footprint of the aircraft may differ.**

Keywords - aerial situation image; unmanned aerial vehicles; complete coverage path planning

I. Introduction

The technical advance in the development of miniature unmanned aerial vehicles (UAVs) in the last decade has made unmanned aerial systems more capable and affordable. Hence, nowadays, civil applications are not only conceivable but already reality. Due to the current rate of development and the varied application possibilities of miniaturized unmanned aircraft, an exponential increase in the usage of these systems can be expected.

In this article, our efforts towards the development of a system for rapidly generating high-resolution aerial imagery using UAVs are described. The first contribution of the article is a new algorithm that efficiently generates a flight plan to completely cover a given area of interest. The area of interest may have complex contours and may contain zones that must not be over-flown by the vehicle. The ability to avoid certain parts will be necessary if obstacles are present or access is forbidden. It will also be useful to prevent irrelevant zones from being inspected. The path planning algorithm and initial results were presented in [1].

The second contribution is a methodology for partitioning the area of interest and distribute the workload of the mission among several aircraft with different ranges and sensor footprints (some initial steps were reported in [2] as a work in progress).

The research presented in this paper builds upon Fraunhofer IOSB's continued efforts to enable rescue and emergency forces to take advantage of UAVs' capabilities in an easy and intuitive way. Previous results include the development of an inspection system for generating high resolution up-to-date aerial images [3]. The system uses the payload capacity of one or several UAVs to scan a defined area with high-resolution image sensors and generates an image mosaic from the accumulated single frames.

Such image acquisition capabilities are integrated into AMFIS, a generic mobile ground control station for monitoring and controlling operation of multiple sensors and sensor carriers [4]. AMFIS features different working positions that enable the users to directly steer the vehicles, and to view their location and other information on a moving map. The unmanned vehicles can also be semi-autonomously controlled through point and click commands. The ground control station receives and displays the video streams and other data coming from the deployed sensors.

With the algorithms presented in this paper the system is now able to deal with no-fly zones and can handle different sensor footprints in a multi-vehicle scenario. Additionally, when compared to the algorithm originally in place, the new planning method yields more efficient paths. Three metrics are used to do this comparison: total flown distance, number of turns and number of "jumps" between non-adjacent cells. While the total distance travelled by the UAV is similar in both cases, with the new method, the number of necessary turns is significantly reduced. The number of jumps is slightly incremented, but with no impact on the total travelled distance.

The paper is organized as follows: related work is discussed in Section II. Section III briefly presents potential application scenarios. Section IV details the proposed path planning algorithm and its results are compared with those obtained with the algorithm previously in use. In Section V the proposed approach for the deployment of multiple heterogeneous vehicles is described. In Section VI, the results obtained with real flight tests after integrating the new algorithms into AMFIS are reported. Section VII provides a conclusion and lines of future work.

II. Related work

A taxonomy proposed by Choset divides coverage path planning algorithms into heuristic based algorithms and algorithms based on a cellular decomposition. The latter can rely on an exact, a semi-approximate or an approximate decomposition [5]. Heuristic based algorithms combine heuristics and randomness to drive the exploration process. These methods, that do not require expensive sensors and do not consume much computational resources, can provide a good ratio between cost and performance; however, parts of the area of interest may remain unvisited. Therefore, complete coverage is not guaranteed. Most of complete coverage path planning algorithms implicitly or explicitly adopt cellular decomposition to achieve completeness.

An exact cellular decomposition is a set of non-intersecting regions, each termed a cell, whose union fills the target environment. Typically, the robot can cover each cell using some kind of motion pattern, e.g., back-and-forth movements, and the path planning algorithm decides the order to visit the cells [6][7][8][9].

Semi-approximate algorithms rely on a partial discretization of space where cells are fixed in width but their top and bottom can have any shape [10][11]. The robot moves along these columns and different parts of a complex area are recursively explored in order to achieve completeness.

An approximate cellular decomposition generates a grid based representation of the area of interest. All cells have the same size and shape and their union approximates the target region. Coverage is complete when the robot visits each cell in the grid. The cell size typically depends on the footprint of the robot. This approach fits very well to our application, where the goal is to generate a complete aerial image of an area by combining aerial photos taken at different points. Many algorithms have been developed that fall into this category. Some of them are referenced in the next paragraphs.

Different authors have developed coverage path planning methods based on spanning trees [12][13][14]. These methods generate a continuous path around the spanning tree. This is a very good property for continuous surveillance operations. The nature of the algorithm requires that, if the cell size derived from the camera footprint is D, the area shall be decomposed into cells of size 4D. Different implementations to generate the spanning trees differ regarding computational complexity and quality of the generated results.

Zelinsky et al. proposed a complete coverage path generation method based on distance transforms [15]. With distance transforms each cell is assigned a value that represents the distance to the goal. These values can be used to find the shortest path from a starting point to the goal. Extensions to the distance transform path planning methodology can be used to generate a complete coverage path. One of the extensions proposed by Zelinsky et al. generates many unnecessary turns. An improved version creates a path that tends to follow the contour of the area. Recently, a distance transform based method has been used by Barrientos et al. to obtain optimal paths in the context of agricultural applications [16]. Their algorithm uses a costly backtracking algorithm to compute all coverage path candidates.

The method proposed by Carvalho et al. makes use of several interesting patterns to generate the path [17]. However, the scanning always takes place in the same direction, which would be a disadvantage in some circumstances, for instance, when a L-shaped area needs to be covered.

An approach developed by Choi et al. creates a path that follows a spiral pattern [18]. Because the algorithm has a tendency to propagate the contour corners towards the inner part of the area of interest, such kind of pattern will not be very efficient in terms of the number of turns when the cell grid has contours with many corners.

The grid that represents the area is used in the method proposed by Kang et al. to create a number of rectangular subareas by grouping cells [19]. Then, one of several patterns is applied to each subarea. We believe that this method can work well when the alignment of boundary cells tends to form rectilinear sides. A number of cells may be revisited when moving from the end of one pattern to the start of the next one.

Segor et al. describe a system that is able to use several small UAVs to efficiently obtain a complete aerial image [3]. Each UAV is allocated one subarea to scan. To cover each subarea two candidate paths are generated: one that makes progress by scanning the area column by column, and another one that does the same row by row. The one that yields better results is chosen. A drawback of such approach is that it does not adapt well to situations where a combination of different scan directions would be more beneficial. Besides, the original implementation was not designed to support no-fly zones inside the area of interest. This work, developed at Fraunhofer IOSB, was the starting point of the research presented in this article.

As shown in Section IV, the new path planning algorithm that we are proposing, significantly improves the results of the previous method used in the AMFIS system. The generation of more efficient flight paths does not come at the expense of a significant increase in computation time.

Regarding the use of multiple aircraft for aerial sensing applications, several projects exist that propose solutions for such scenario. In the SkyObserver project a geometric optimal placement algorithm is used to distribute the unmanned aircraft inside a given area [20]. In the AirShield and AVIGLE projects similar techniques are used [20][21][22]. However, in this case a more holistic approach that simultaneously considers spatial coverage optimization, the mobility strategy to explore the area, and communication awareness is followed.

In the COMETS project, the area of interest is first decomposed into non-intersecting regions taking into account UAV’s relative capabilities and initial locations [6]. Afterwards, the resulting areas are assigned among the UAVs that will cover them using a zigzag pattern.

Finally, the problem of creating an aerial image of a given area is also addressed in the cDrones project [24][25]. Like in our case, the proposed solution is based on a grid approximation of the area of interest. A genetic algorithm is used to compute the flight path. While the system is able to use multiple aircraft, the method for distributing the workload is not described.

Decentralized planning methods as in SkyObserver, AirShield and AVIGLE try to maintain connectivity between

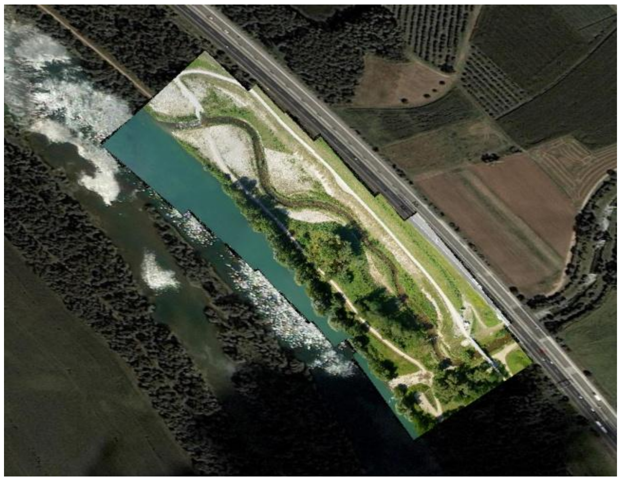


Figure 1. Photomosaic of Biotope at the Rhein River.

nodes. As a result, constraints imposed on the trajectories make complete coverage harder to achieve. We propose a practical approach where each aircraft executes a pre-assigned path. With the exception of cDrones, the referenced projects do not provide support for no-fly zones. Finally, most approaches do not consider the possibility to perform the mission with heterogeneous platforms.

III. APPLICATION SCENARIOS

The security feeling of our society has significantly changed during the past years. Besides the risks arising from natural disasters, there are dangers in connection with criminal or terroristic activities, traffic accidents or accidents in industrial environments. Especially in the civil domain in case of big incidents there is a need for a better data basis to support the rescue forces in decision making.

The search for buried people after building collapses or the clarification of fires at big factories or chemical plants are possible scenarios addressed by our system.

Many of these events have very similar characteristics. They cannot be foreseen in their temporal and local occurrence so that situational in situ security or supervision systems are not present. The data basis for decision making is rather thin and therefore the present situation is very unclear to the rescue forces at the beginning of a mission. Exactly in such situations it is extremely important to understand the context as fast as possible to initiate the suitable measures specifically and efficiently.

An up-to-date aerial image can be a valuable additional piece of information to support the briefing and decision making process of the first responders.

Helicopters or supervision airplanes that can supply this information are very expensive or may be unavailable. High-resolution pictures from an earth observation satellite could also be a good solution in many cases. But, under normal circumstances, these systems will not be available in time or they may not be able to deliver good pictures because of clouds or smoke. A small, transportable, fast and easily deployable system that is able to produce results with higher spatial and temporal resolution is proposed to close this gap.

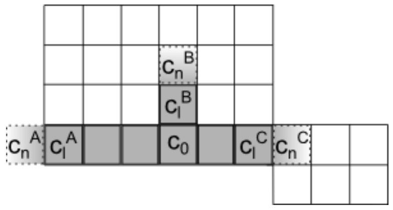


Figure 2. Stride formation starting at cell c0.

The aerial inspection tool described in this paper can provide the lacking information by creating an overview of the site of the incident in a very short time. The application can be used by first responders directly on site with relative ease. The results provide a huge enhancement to the available information.

Many other applications are also possible: support to fire-fighting work, clarification of debris and the surroundings after building collapses, search for buried or injured people, inspection of large objects, or for documentation and perpetuation purposes, as for example, of protected areas and biotopes (see Fig. 1).

IV. PATH PLANNING

Once the area of interest has been identified and its grid approximation has been computed, a flight path will be generated for each unmanned vehicle participating in the mission. Two steps are required during this process: first, the area is partitioned according to the capabilities of each aircraft. Next, a flight path to cover each of the subareas is computed. These flight paths contain sequences of waypoints where pictures need to be taken in order to completely cover the area.

In this section, the path planning algorithm is described in detail. The algorithm will be applied to the whole grid if there is a single UAV, or to each one of the subareas in a multi-UAV scenario. Also in this section, the results of its application to several different areas are presented. These results are compared to the ones obtained with the previous algorithm used in AMFIS.

A. Path Planning Algorithm

The path planning algorithm tries to generate the longest possible straight flight segments. We call each one of these segments a stride. A stride is defined as a sequence of consecutive adjacent cells without turns. To compute a stride of max length the following rules are used:

- The stride starts at the current cell.
- The direction of the stride is determined by its starting point and the neighbor under consideration.
- A stride contains no turns.
- A number of conditions, explained below, determine where the stride ends.

We define $L(c)$ as the number of visited neighbor cells and area limits located orthogonally to the stride direction at cell c. Therefore, assuming a stride direction from left to right, the value of $L(c)$ will be 2 if the cells located immediately on top and below c are marked as visited or fall outside the area of interest. Conversely, $L(c)$ will be 0 if both

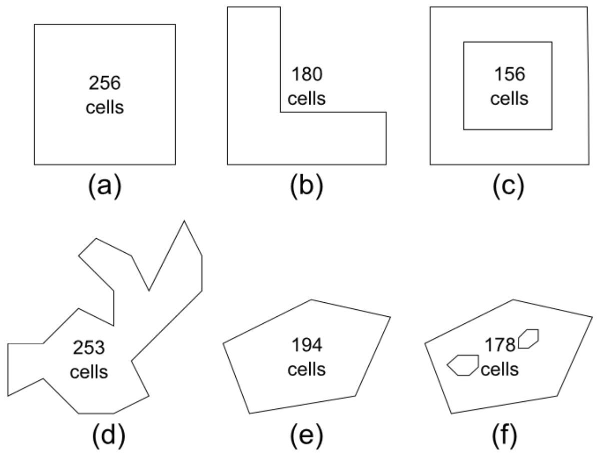


Figure 3. Contours, with their number of cells, used in the tests.

cells are inside the area of interest and still need to be covered.

Being c_0 the first cell of the stride, c_l the current last cell of the stride, and c_n a potential next cell in the stride direction, addition of c_n to the stride is subject to the following conditions:

- C_n is not added to the generated path if it is already marked as visited or if it falls outside the boundaries of the area.
- If the previous condition does not hold, and $L(c_l) = 2$, c_n is always added to the stride, because it is the only unvisited cell that can be reached from c_l.
- If $L(c_l) \neq 2$, any of the following conditions will prevent addition of c_n to the stride: (1) $L(c_n) = 0$; (2) $L(c_n) \neq L(c_0)$; or (3) $L(c_n) = 1$ but the limit at c_n is positioned opposite to the limit found at c_0. The purpose of condition (1) is to stop when the path is not following an area limit or a "wall" formed by cells already in the path. Conditions (2) and (3) dictate that stride formation will also stop when the limits of c_n differ from the limits of c_0.

To clarify the previous points, stride formation is illustrated in Fig. 2. Starting at c_0, there are three alternative strides that can be selected. Stride A (left of c_0) ends when an area limit is reached. Stride B (up) ends because $L(c_n^B) = 0$. Note that it would be possible to extend the stride because there are unvisited cells in that direction, but this would eventually lead to the partition of the area into two disconnected regions. Finally, in stride C (right of c_0), both c_0 and c_n^C have an area limit located orthogonally to the stride direction. Therefore $L(c_0) = L(c_n^C) = 1$, however, since the area limits of these cells are located at opposite sides (below in the case of c_0 and above for c_n^C), c_n^C is not added to the stride.

The algorithm for generating the complete coverage path works as follows:

1. Set the current cell to the initial cell.
2. Find all unvisited neighbor cells of the current cell (between 0 and 4 cells are returned).
3. Generate the longest possible stride in the direction of each unvisited neighbor cell.
4. Select the longest stride.

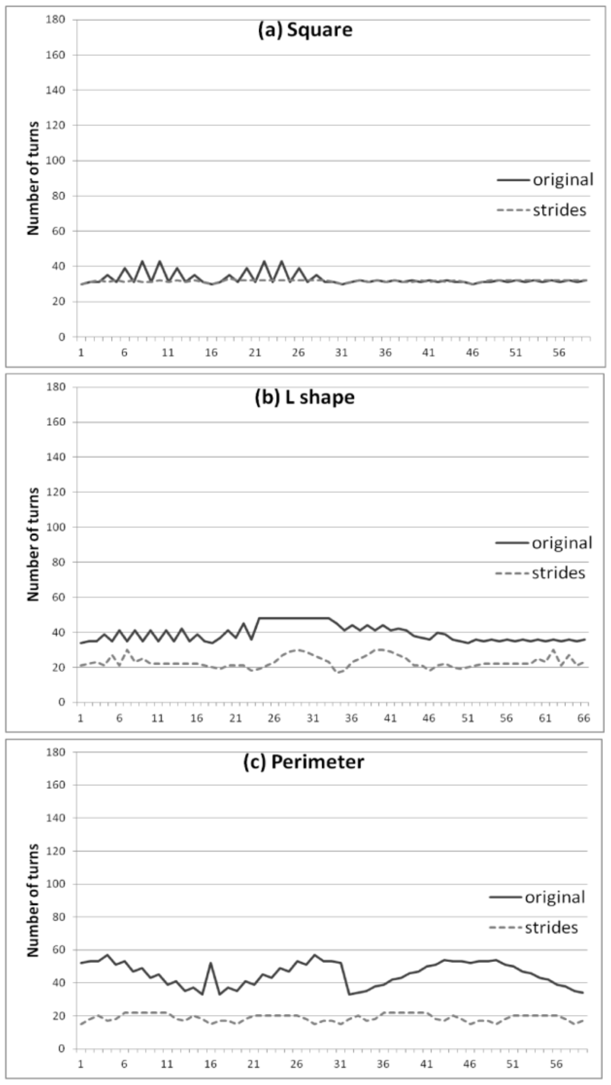


Figure 4. Number of turns starting at each contour cell of a, b and c.

5. Add all cells of the stride to the path and mark them as visited.
6. Set the current cell to the last cell of the stride.
7. Repeat starting at point 2 until all cells have been visited.

In the example presented in Fig. 2, stride A, with four cells, would be selected over the B and C alternatives, which respectively have two and three cells.

When all alternative strides have length two (c_0 plus a neighbor cell), some heuristics are used to perform the stride selection. These heuristics prioritize the selection of (1) the neighbor cells with a higher value for $L(c_n^C)$, (2) the ones located in the contour of the area, and (3) the ones that lead to a longer stride in the next step. These heuristics have been chosen after extensive testing with different area shapes. Heuristics (1) and (2) promote the selection of a path that moves alongside other visited cells or the contour of the area.

Sometimes it is inevitable to partition the area, creating two, or more, subareas of disconnected unvisited cells. When

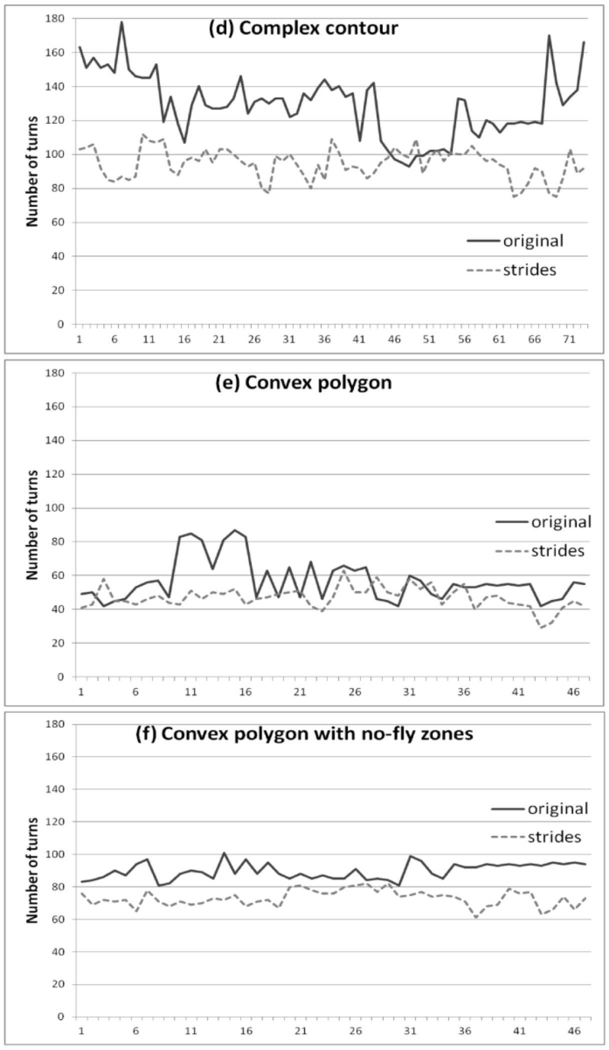


Figure 5. Number of turns starting at each contour cell of d, e, and f.

such situation occurs, the algorithm chooses to visit the cells of the smallest subarea first.

Another situation that needs to be addressed occurs when a dead end is reached, i.e., when there are no unvisited valid cells next to the current cell. In this case, our solution computes paths between the current cell and each one of the unvisited cells adjacent to cells already in the flight path. These paths are computed using distance transform path planning as described by Zelinsky et al. in [15]. After all the alternatives have been computed, the shortest path is selected and its cells are added to the complete coverage path. This same method can be used to compute a path to the landing point.

The distance wave propagation used to compute the shortest path (see [15]), is also used to determine if the area has been partitioned. The main idea behind this procedure is that, after performing the propagation of the distance wave, if any cells remain that have not been assigned a value they must be located in a different partition.

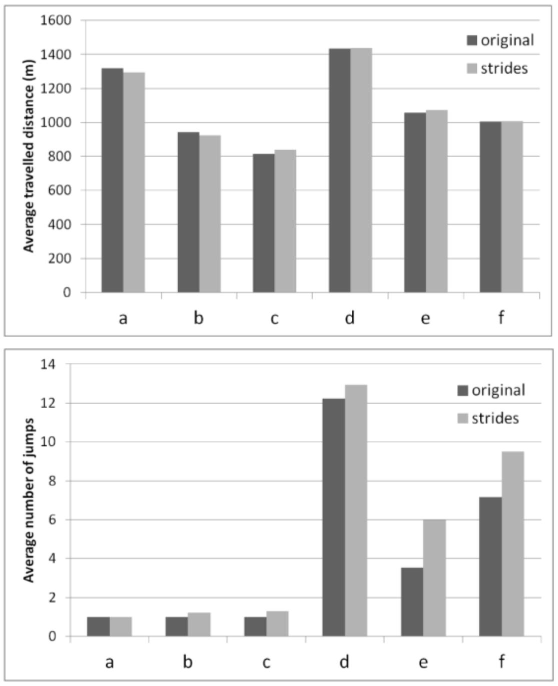


Figure 6. Average travelled distance (top) and average number of jumps (bottom).

One final step performs some clean-up on the generated path. Those sequences of revisited cells that connect two adjacent cells are removed from the path. In this way, the UAV transitions directly from a given cell to one of its neighbors and unnecessary repeats are prevented. Since the number of times each cell is visited is known, the implementation of the clean-up step is straightforward. For each cell C_i of the flight plan, the algorithm checks if the rest of the path contains a cell *Cj* that is adjacent to C_i. If all cells in between are visited more than once, they can safely be removed.

B. Results

The proposed algorithm has been tested with areas of different shape. In this section, the results obtained with the areas showed in Fig. 3 are presented. The new algorithm has been compared with the original algorithm of the photo flight tool. The metrics used are the number of turns, the travelled distance, and the number of jumps, which are the transitions between non-adjacent cells. Some considerations need to be taken into account to analyze the results:

1. In a situation where all neighbor cells next to the current position have been visited, but coverage is not complete, the original algorithm didn't provide a safe path to fly from the current position to the next free cell. For this reason the number of jumps between non-adjacent cells is compared instead of the number of revisited cells.
2. The original algorithm was not designed to handle no-fly zones inside the area of interest. Nevertheless, if such an area is provided as input, it is able to generate a complete coverage path.

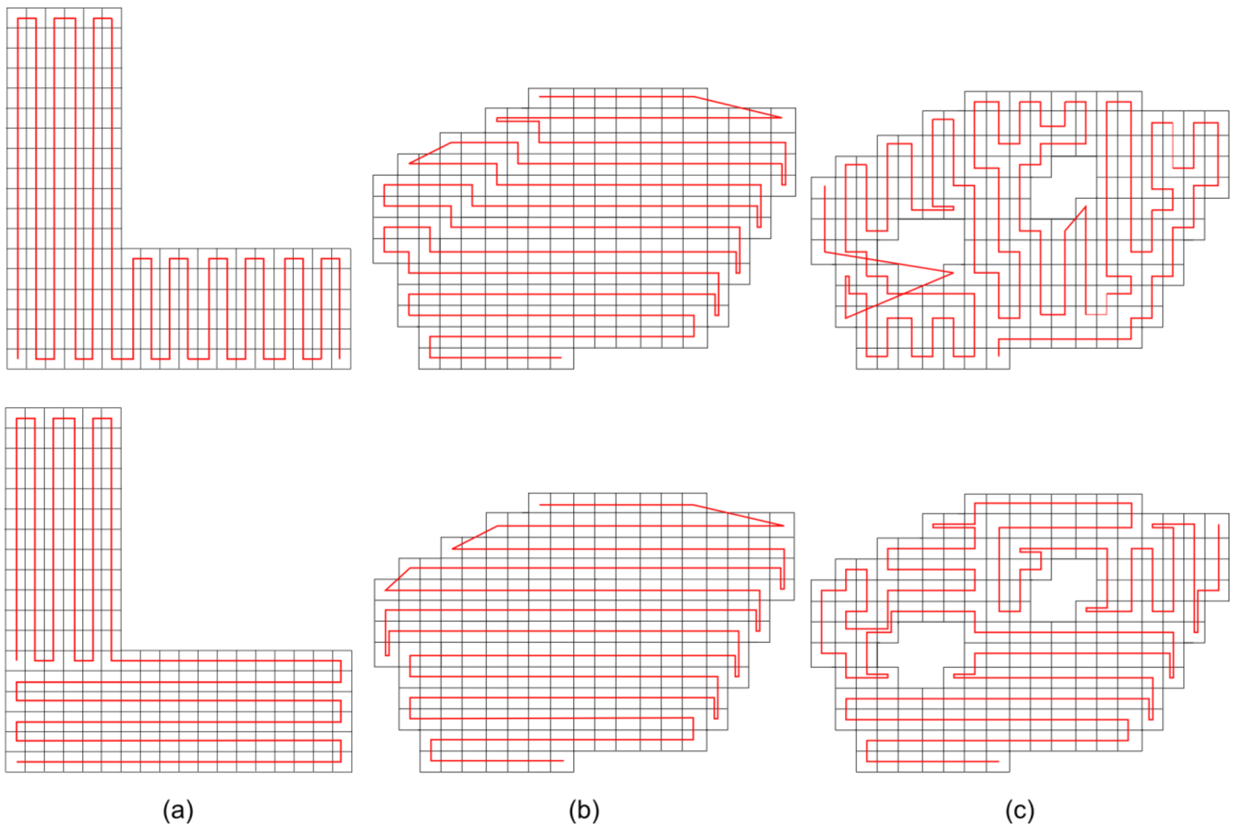


Figure 7. Complete coverage path generated for some example contours with both original (top) and strides based algorithm (bottom). In figure c the transitions between non-adjacent cells are also included.

3. The computed distance corresponds to the path length plus the distance between the last and the first cell of the path.

In Figs. 4 and 5, the number of turns obtained running both the original and the new strides based algorithms are displayed. The algorithms have been run starting at each contour cell of the example areas in Fig. 3. As it can be observed, the new algorithm provides better results with all tested areas.

Fig. 6 displays the average travelled distance (top) and average number of jumps (bottom) obtained, with both algorithms, for each area. It can be seen that, although the number of jumps is slightly increased in some cases, this increase has almost no impact on the average travelled distance.

To understand the reasons that lead to an improvement in the number of turns, we now compare the complete paths of the areas b and e of Fig. 3. In Fig. 7a, it can be seen that one source of improvement is the ability of the new algorithm to use different scan directions in different parts of the area of interest. Another source of improvement (see Fig. 7b), comes from the fact that the new algorithm is better at getting rid of contour corners, avoiding its propagation into the inner parts of the area (see Fig. 7b).

Finally, in Fig. 7c (bottom), the complete path generated for a complex area with no-fly zones is shown. When there are no unvisited neighbors, a safe path to reach the next free cell is computed. Thus, the complete coverage path does not contain jumps between non-adjacent cells. The generated path can be contrasted with a path generated by the original algorithm (top), which was not really designed to cope with holes in the area, and does not provide a mechanism to generate a safe path between non-adjacent cells.

It should be noted that the path planning algorithm always operates on a grid where each cell is a square. However, the actual footprint could also be a rectangle whose sides differ in length. In that case, the size of the square cells will be determined by the longest side of the rectangle. Once the path along the grid has been computed, an additional step is needed to compute the list of waypoints required to take all the pictures according to the actual footprint. The orientation of the sensor carrier will be determined in accordance to the flying direction.

V. PLANNING FOR MULTIPLE HETEROGENEOUS VEHICLES

In a multi-UAV scenario, the application of the path planning algorithm is preceded by a partitioning of the area of interest to distribute the workload of the mission among the available vehicles. In this section, the method used to perform such partitioning is described.

A. *Workload Distribution*

The partitioning algorithm needs to fulfill a number of requirements. It needs to provide support for using platforms with different capabilities regarding range of the vehicles, speed, sensor size, and sensor resolution. It must be able to

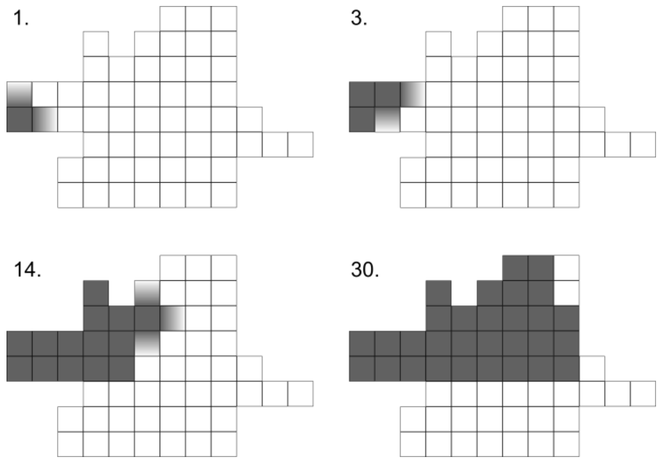


Figure 8. A flood-fill like algorithm is applied to assign cells to a partition.

cope with areas with concave contours, possibly containing holes. Pieces that facilitate the path planning should be generated, i.e., the generation of subareas with unnecessary corners should be avoided. Finally, in order to provide a rapid response in emergency situations, the method for partitioning the area needs to be fast.

To partition the area of interest, an initial method already implemented in the AMFIS ground control station has been extended. As the path planning algorithm, the partitioning method relies on a grid approximation of the area of interest.

The first step consists in computing the percentage of the area of interest that should be covered by each vehicle. However, the actual length of a computed flight path does not only depend on the number of cells that need to be covered. Aspects such as shape of the area, sensor footprint and actual decisions made during the planning process also have an impact on the traveled distance. Since it is not possible to determine the flight length before actually computing the flight plan, the percentage of the area that will be allocated to each aircraft is based on a rough estimation that takes into account the sensor footprint of each aircraft.

The percentage values are used to determine the number of cells of the grid required for each subarea. To perform the partitioning an initial cell is selected and a flood-fill like algorithm is applied to extend the subarea until the desired number of cells is reached. The process, depicted in Fig. 8, prioritizes the selection of adjacent cells with a higher number of neighbor cells in the same subarea. Cells outside the area of interest or in holes are ignored.

Once an initial solution has been computed, remaining cells that have not been allocated (see Fig.9) are assigned to an adjacent area. This step is followed by a redistribution of cells to obtain the desired number of cells in each subarea.

B. *Dealing with Multiple Sensor Footprints*

If all aircraft involved in the mission have a sensor footprint of the same size, a single grid can be used to partition the area and to plan complete coverage flight paths. In Fig. 10, the results of partitioning a given area to divide the work between three aircraft are displayed (10.a), together with the flight paths generated by the path planning algorithm (10.b).

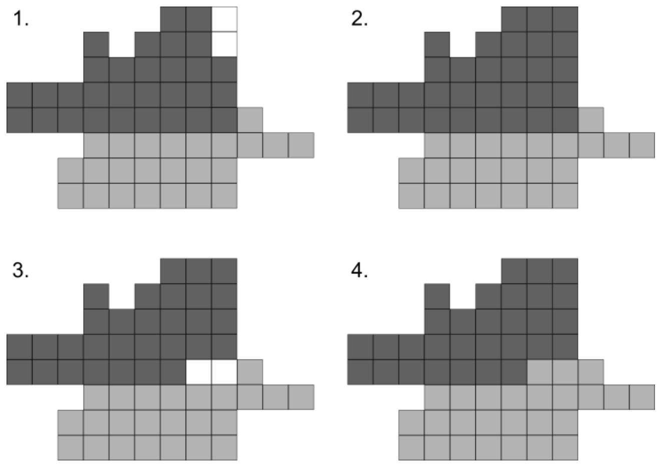


Figure 9. Initial partition may be rearranged to properly allocate all cells.

If the sizes of the sensor footprints differ, using a single grid is no longer possible. In the following paragraphs the method used to deal with multiple footprints is described.

The first step consists in creating a base grid of square cells to partition the area and distribute the work between the different vehicles. Assuming that there are n vehicles available to perform the mission, with w_i and h_i respectively representing the width and the height of the i^{th} vehicle's footprint, the side length s of the grid cells is computed as:

$$s = Min_{i=1}^{n}(Max(w_i, h_i)).$$

It should be noted that, for convenience, we use the word footprint to refer to an area that is actually smaller than the area on the ground surface captured by the sensor. The use of an area smaller than the real footprint is done in purpose to introduce an overlap between the images that will facilitate the stitching process. This overlap will also be helpful to mitigate inaccuracies of the positioning system.

Once the base grid has been created, the partitioning is performed as previously described. The resulting subareas may not be directly used by the path planning algorithm due to the mismatch between the cell size of the grid and the footprint of the different aircraft. A new grid with a cell size proportional to the sensor footprint will be created for each aircraft. The cells that will be covered in each new grid are those that overlap with the subarea assigned to the aircraft in the base grid (see Fig. 11.b).

For efficiency and safety reasons overlaps between areas allocated to different aircrafts should be minimized. The next step of the process, the generation of waypoints, tries to minimize these redundancies.

The main goals of the waypoint generation step are:

1. Convert the paths computed as cells on a grid to actual flight plan waypoints: To minimize the distance between consecutive waypoints, the on-board sensors will be oriented with their longest side orthogonal to the direction of movement. The length of the shortest side is used as the distance between consecutive waypoints.
2. Prevent overlapped zones between subareas from being visited multiple times: If a location has been

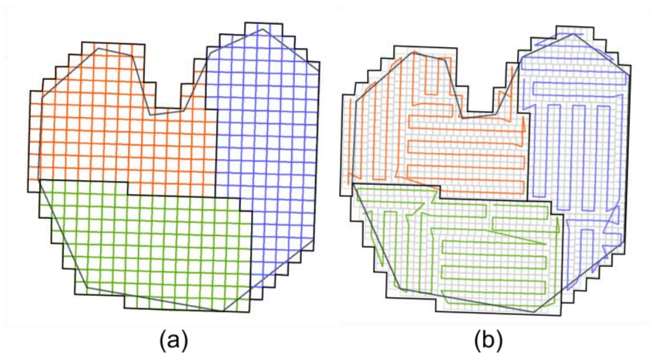


Figure 10. Area partitioning (a) and path planning (b) with same cell size.

photographed by one of the aircraft, waypoints for taking additional pictures of the same place with another aircraft are not generated.

Having multiple aircraft visiting nearby locations has safety implications. To avoid collisions between the unmanned vehicles several approaches are possible. One way to solve this problem would consist in improving the planning phase so that the probability of multiple vehicles being at the same place at the same time is minimized. Another possibility would be to add the necessary sensory and computing capabilities to the vehicles to autonomously detect and deal with possible conflicts. Finally, an effective and practical approach consists in ensuring that the vehicles fly at different heights with enough separation. Since the presented planning method is able to deal with different footprints, this approach can be easily implemented in our system.

VI. EXPERIMENTAL RESULTS

The platform chosen to perform the experiment is the Falcon 8 octocopter from Ascending Technologies. The Falcon 8 comes equipped with an autopilot and a GPS sensor that enable autonomous flight. A camera installed on a stabilized mount can be automatically triggered every time a waypoint is reached. While the high resolution pictures need to be recovered after landing, the system is able to provide a continuous video stream that can be displayed on the ground control station. The Falcon 8 is a lightweight system of 2.2 kg maximum take-off weight that provides up to 20 minutes flight time.

The partitioning and planning algorithms presented in this article have been integrated into the AMFIS system. The mobile ground control station provides tools for designing the mission, supervising its execution and analyzing the obtained results.

With the flight planning tool (partially shown in Fig. 12) the user will perform the following steps:

1. Mark the region of interest on a map.
2. Select the unmanned aircraft that will be used in the mission.
3. Set the flight altitude for each aircraft, or define the desired resolution.
4. Compute a flight path for each aircraft.

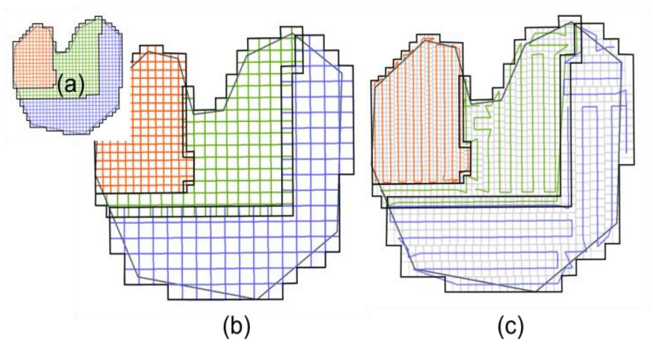


Figure 11. Area partitioning (a+b) and path planning (c) with different cell size.

5. Send the obtained paths to the different vehicles.

Once the previous steps are completed, the vehicles can be commanded to autonomously perform the mission. After the recovery of the pictures, the same photo flight tool can be used to stitch them together and provide a complete high resolution image of the area.

Our image acquisition system was tested in the Karlsruhe facilities of Fraunhofer IOSB. For safety reasons, the flight altitude was set to 100 meters. At such altitude the footprint of the camera has an approximate size of 120 x 90 square meters. Therefore, with only a few pictures the available area for performing the tests, which approximately measures 17.000 square meters, would have been covered. To force a smaller footprint, the flights were planned at a height below the safe altitude. Once the flight paths had been computed, we took advantage of the editing functions of the ground control station to set the altitude of all waypoints to a safe value. Such scenario results in a big overlap between pictures that facilitates the stitching process, but does not affect the path planning aspect, which was the focus of our interest.

Our tests consisted in three flights. The first flight covered the whole area with a single vehicle. The second and third flights were planned assuming that two vehicles would be flying simultaneously. The flights were planned using different altitudes to prevent collision between the vehicles. The use of different flight altitudes results in different footprint sizes, which our flight planning tool proved to be able to handle. To prevent any risks, the second and third flights were actually performed sequentially using the same vehicle. This also meant that a single emergency pilot would suffice. The number of taken pictures, the horizontal flown distance and the duration of each flight can be found in Table I. Distance and flight duration were measured between the first and the last waypoint.

Table I. Number of pictures, horizontally travelled distance and flight time.

Flight	Pictures	Distance (meters)	Time (mm:ss)
1st	18	420	03:26
2nd	16	320	02:15
3rd	13	240	01:57

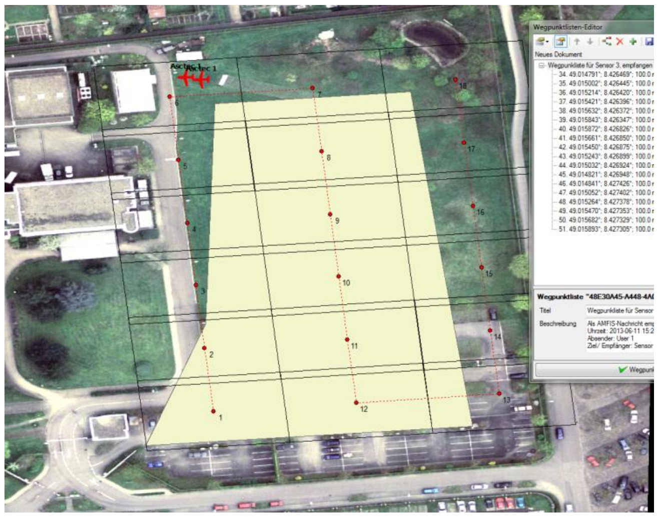


Figure 12. Path planning tool detail.

The results of the experiments were highly satisfactory. We were able to successfully plan the mission with the path planning tool, send the flight plans to the vehicle, share them with other components of the ground control station, and finally execute the mission without incidents. Human intervention was only required to perform the take-off and landing operations because the Falcon 8 platform does not provide support to perform these operations automatically. The aerial image obtained after combining all individual pictures is shown in Fig. 13.

The main issues encountered during the tests relate to the use of a grid approximation to represent the area of interest. Sometimes most of the area covered by boundary cells falls outside the area of interest. While these cells are necessary to guarantee complete coverage, this situation can lead to inefficiencies that are more evident when the number of boundary cells is high in relation to the total number of cells.

Moreover, the presence of cells with large parts of their area lying outside of the region of interest can result in the aerial vehicle flying outside the region's boundaries. This can be a problem if such boundaries have been defined to avoid obstacles. In our tests we wanted to keep a safe distance from the buildings and also avoid crossing the limits of the institute's facilities.

Since the AMFIS ground control station provides support for editing the flight plans, such functions can be used to make manual adjustments and prevent the vehicle from flying over undesired zones. While this could be acceptable in some cases, it is necessary to improve the planning tool to provide a more general solution. We believe that this problem can be tackled with the addition of a new processing step that would take the flight plan as input and generate a modified version were all waypoints would lie inside the area of interest, possibly incrementing the overlap between the pictures.

VII. CONCLUSION AND FUTURE WORK

In this paper, we introduced our most recent work targeted at the development of a system to enable fast generation of aerial image mosaics. We believe that such system can provide highly valuable information in emergency situations. The proposed partitioning and path planning algorithms, which are able to generate efficient solutions in a short time, are core elements of the system. The method for solving the area partitioning problem in the presence of multiple vehicles is able to cope with different sensor footprints. During the path planning a criterion that prioritizes the selection of long straight segments is applied. Such approach results in the generation of flight paths with a reduced number of turns. Fast moving aircraft will particularly benefit from having to perform less turns. Its ability to scan the area in different directions and the fact that it does not rely on pre-defined patterns make the proposed planning algorithm suitable to generate complete coverage paths for complex contours, which may contain holes.

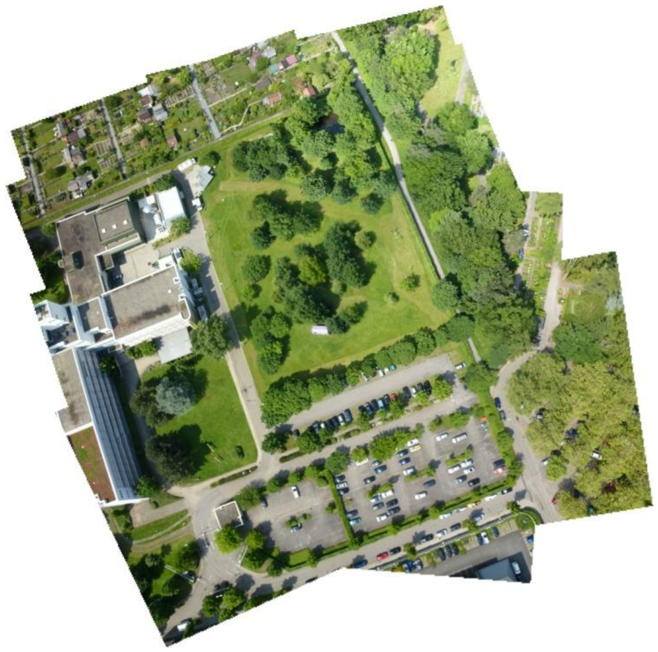


Figure 13. Mosaic obtained after stitching individual pictures.

The partitioning and planning methods have been integrated into the AMFIS ground control station and the results of experimental flights are reported. The system is appropriate for maneuverable vehicles, such as multi-rotor aircraft.

There are several aspects that require further work. More extensive and realistic tests, with bigger and more complex areas should be performed. For efficiency and safety reasons, the planning tool should be improved so that the aircraft do not cross the boundaries of the region of interest. The system should also automatically detect and provide solutions to situations where multiple flights are necessary due to range limitations of the vehicles. Another interesting extension would be to adapt the system to accommodate fixed-wing aircrafts, which are not able to perform sharp turns. Finally, it would be very interesting to explore the operational aspects and study how the aerial image acquisition system should be integrated into the decision making processes during emergency situations.

Acknowledgment

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

References

[1] E. Santamaria, F. Segor, I. Tchouchenkov, and R. Schönbein, “Path Planning for Rapid Aerial Mapping with Unmanned Aircraft Systems”, Proceedings of the Eighth International Conference on Systems ICONS, 2013.

[2] E. Santamaria, F. Segor, and I. Tchouchenkov, “Rapid Aerial Mapping with Multiple Heterogeneous Unmanned Vehicles”, Proc. of the 10th International Conference on Information Systems for Crisis Response and Management ISCRAM, 2013.

[3] F. Segor, A. Bürkle, M. Kollmann, and R. Schönbein, “Instantaneous Autonomous Aerial Reconnaissance for Civil Applications - A UAV based approach to support security and rescue forces”, 6th International Conference on Systems ICONS, 2011, pp. 72-76.

[4] F. Segor, A. Bürkle, T. Partmann, and R. Schönbein, “Mobile Ground Control Station for Local Surveillance”, Proc.of the Fitth International Conference on Systems ICONS, 2010.

[5] H. Choset, “Coverage for robotics - A survey of recent results”, Annals of Mathematics and Artificial Intelligence vol.31, 2001, pp.113-126.

[6] I. Maza and A. Ollero, “Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms”, in Distributed Autonomous Robotic Systems vol. 6, R. Alami and R. Chatila and H. Asama, Eds. Springer Japan, 2007, pp. 221-230.

[7] H. Choset, “Coverage of Known Spaces: The Boustrophedon Cellular Decomposition”, Autonomous Robots vol. 9, 2000, pp.247-253.

[8] R. Mannadiar and I. Rekleitis, “Optimal coverage of a known arbitrary environment”, IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 5525 -5530.

[9] W. Huang, “Optimal line-sweep-based decompositions for coverage algorithms”, IEEE International Conference on Robotics and Automation (ICRA) vol.1, 2001, pp. 27 - 32.

[10] S. Hert, S. Tiwari, and V. Lumelsky, “A terrain-covering algorithm for an AUV”, Autonomous Robots vol.3, 1996, pp.91-119.

[11] V. Lumelsky, S. Mukhopadhyay, and K. Sun, “Dynamic path planning in sensor-based terrain acquisition”, IEEE Transactions on Robotics and Automation 6(4), 1990, pp.462 -472.

[12] Y. Gabriely and E. Rimon, “Spanning-tree based coverage of continuous areas by a mobile robot”, Annals of Mathematics and Artificial Intelligence vol. 31, 2001, pp.77-98.

[13] P. J. Jones, “Cooperative area surveillance strategies using multiple unmanned systems”, PhD thesis, Georgia Institute of Technology, 2009.

[14] M. Weiss-Cohen, I. Sirotin, and E. Rave, “Lawn Mowing System for Known Areas”, International Conference on Computational Intelligence for Modelling Control Automation, 2008, pp. 539 -544.

[15] A. Zelinsky, R. Jarvis, J. C. Byrne, and S. Yuta, “Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot”, International Conference on Advanced Robotics, 1993, pp. 533—538

[16] A. Barrientos, J. Colorado, J. del Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, “Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots”, J. Field Robot. 28(5), 2011, pp. 667--689.

[17] R. De Carvalho, H. Vidal, P. Vieira, and M. Ribeiro, “Complete coverage path planning and guidance for cleaning robots”, ' IEEE International Symposium on Industrial Electronics, 1997. ISIE 97, vol. 2, pp. 677 -682.

[18] Y.-H. Choi, T.-K. Lee, S.-H. Baek, and S.-Y. Oh, “Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform”, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009, pp. 5788 -5793.

[19] J. W. Kang, S. J. Kim, M. J. Chung, H. Myung, J.H. Park, and S. W. Bang, “Path Planning for Complete and Efficient Coverage Operation of Mobile Robots”, International Conference on Mechatronics and Automation, 2007. ICMA 2007', pp. 2126 -2131.

[20] T. Passenbrunner and L. del Re, “SkyObserver: Decentralized, real-time algorithm for deployment of a swarm of Unmanned Aircraft Systems”, Proceedings of the 9th IEEE International Conference on Control and Automation, 2011.

[21] H. Daniel, S. Rohde, N. Goddemeier, and C. Wietfeld, “Cognitive Agent Mobility for Aerial Sensor Networks”, IEEE Sensors Journal, 2011, 11, 2671 -2682.

[22] S. Rohde, N. Goddemeier, C. Wietfeld, F. Steinicke, K. Hinrichs, T. Ostermann, J. Holsten, and D. Moormann, “AVIGLE: A system of systems concept for an avionic digital service platform based on Micro Unmanned Aerial Vehicles Systems”, IEEE International Conference on Man and Cybernetics, 2010.

[23] K. Daniel, B. Dusza, A. Lewandowski and C. Wietfeld, “AirShield: A system-of-systems MUAV remote sensing architecture for disaster response”, 3rd Annual IEEE Systems Conference, 2009.

[24] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner, “Networked UAVs as aerial sensor network for disaster management applications” , e & i Elektrotechnik und Informationstechnik, Springer, 2010, 127, 56-63.

[25] M. Quaritsch, R. Kuschnig, H. Hellwagner, B. Rinner, “Fast Aerial Image Acquisition and Mosaicking for Emergency Response Operations by Collaborative UAVs”, Proceedings of the 8th International ISCRAM Conference, 2011.

Time Series Prediction with Automated Periodicity Detection

Michael Schaidnagel
School of Computing
University of the West of Scotland
Paisley PA1 2BE, UK
Email: Michael.Schaidnagel@web.de

Fritz Laux
Faculty of Computer Science
Reutlingen University
D-72762 Reutlingen, Germany
Email: fritz.laux@reutlingen-university.de

***Abstract*—When forecasting sales figures, not only the sales history but also the future price of a product will influence the sales quantity. At first sight, multivariate time series seem to be the appropriate model for this task. Nonetheless, in real life history is not always repeatable, i.e., in the case of sales history there is only one price for a product at a given time. This complicates the design of a multivariate time series. However, for some seasonal or perishable products the price is rather a function of the expiration date than of the sales history. This additional information can help to design a more accurate and causal time series model. The proposed solution uses an univariate time series model but takes the price of a product as a parameter that influences systematically the prediction based on a calculated periodicity. The price influence is computed based on historical sales data using correlation analysis and adjustable price ranges to identify products with comparable history. The periodicity is calculated based on a novel approach that is based on data folding and Pearson Correlation. Compared to other techniques this approach is easy to compute and allows to preset the price parameter for predictions and simulations. Tests with data from the Data Mining Cup 2012 as well as artificial data demonstrate better results than established sophisticated time series methods.**

***Index Terms*—sales prediction, multivariate time series, periodicity mining**

I. Introduction

Time series capture the development of given values over a uniform time interval. There are many areas in which this kind of data can appear: power consumption data of different housing areas per month, heartbeat rate of a patient per minute, hourly weather data or also product sales per day. In this article we will focus mainly on prediction of sales time series in order to keep the main thread examples consistent. However, this is not a limitation since the underlying patterns, which are used in our algorithms, can occur in all of above's application areas.

This work is based on our findings from [1], in which we firstly applied an proposed algorithm named F_r on sales data. The algorithm is characterized by its ability to use the price as a input variable as well as the adaption of a hidden periodicity. Sales prediction is an important goal for any time series based analysis [2], [3]. The task consists of forecasting sales quantities given the sales history. This can be achieved by extending the time series into the future.

The extrapolation of the time series into the future is determined by the underpinning time series model [4]. If this model is not well supported by the empirical data it is likely that the accuracy of the forecast is low. So the challenge is to find data from "similar" situations (e.g., in terms of time and price). If a major sales factor like the product price changes, a model solely based on previous sales will lead to wrong forecasts. Therefore, it is important to include the price as parameter into the model in addition to the sales history.

Standard solutions for this problem need to be provided with a long history of sales with sufficient data to validate the model and to correlate the sales data with the variable product price. The mathematical tools of choice for analyzing multiple time series simultaneously are multivariate statistical techniques like Vector AutoRegressive (VAR) models [5], [6] or such as the Vector ARIMA (AutoRegressive Integrated Moving Average) [7]. The model parameters are estimated with least square or Yule-Walker functions [5]. The accuracy of the estimator depends on the number of observations and its degree of correlation.

To illustrate the process consider an excerpt from the Data Mining Cup 2012 dataset (Table I, http://www.data-mining-cup.de/en/review/dmc-2012):

TABLE I
Sample Data, Data Mining Cup 2012

day	Prod#	price	quantity
1	1	4.73	6
1	2	7.23	0
1	3	10.23	1
1	4	17.90	0
...	...	...	...
1	570	7.91	0
2	1	4.73	12
2	2	7.23	1
...	...	...	...
42	569	9.83	2
42	570	7.84	0
43	1	5.35	?
43	2	7.47	?
...	...	...	...
43	570	7.84	?
...	...	...	...
56	570	8.12	?

The information provided comprises a collection of 570 products whose history of sales and prices are given over a period of 42 days. The task was to predict the sales quantities for the next 14 days where the daily sales price was preset. The majority of products produced only low quantity sales. Comparisons with other sales data showed a similar distribution [8], [9] which indicates that the sample is typical for larger collections. When we tried to predict the future sales with commercial ARIMA products we experienced a low prediction quality with a relative accuracy of only 47%.

The disappointing results from professional tools implementing ARIMA encouraged us to look for a simpler and better prediction model. Thereby we assumed that the future price is causally influenced and should not be treated as stochastic variable. Second, we assumed that it is helpful to filter out cyclic behavior from the "white noise".

In the next section follows a discussion of related work and we contrast it with our contribution. The rest of the paper is structured as follows: The research problem will be described formally in Section III, which is followed by a description of data profiles under investigation (Section IV). In Section V, we present our parametrized time series algorithm that predicts sales volumes with variable product prices and low data support. This algorithm can benefit from a inherent (i.e., hidden) periodicity within the given data. The periodicity calculation method we used is further described in Section VI. The following Section VII gives a description of the technical framework for the implementation of the prototype. The results are discussed in Section VIII and compared with standard methods found in commercial products like ARIMA. Based on these experiences we draw conclusion in the last section.

II. Related Work

Adaptive correlation methods for prognostic purposes have been proposed early in the 1970^{th} by Griese [10] and more specifically as AutoRegressive Moving Average (ARMA) method by Box and Jenkins [11]. As ARMA is constrained to a stationary stochastic process the ARIMA is of more practical use as it can handle time series with a linear trend and is therefore widely implemented.

The idea behind ARMA and ARIMA is that the model adapts automatically to a given history of data. A natural extension is to include other influential factors beside the prognostic value itself. This leads to multivariate models, namely Vector Autoregressive (VAR) models [7]. The development of the model was influenced and motivated by critiques of Sims [12] and Lucas [13]. In essence, their statement is: every available data is potentially correlated.

If the model is extended to cover the influence from correlated data this leads to a vectorial stochastic model $(\mathbf{X}_t(\pi, \pi_r))$[1] that allows not only the serial time dependence t of each component but also the interdependence of products π and product prices π_r. To estimate the parameters of such a multivariate ARMA process the following Equation has to be solved [14], [5]:

$$\Phi(L)X_t(\pi, \pi_r) \quad = \quad \Theta(L)Z_t \tag{1}$$

where L denotes the backshift (lag) operator and

$$\Phi(x) \quad := \quad I - \Phi_1 x - \Phi_2 x^2 - \ldots - \Phi_p x^p \tag{2}$$

$$\Theta(x) \quad := \quad I + \Theta_1 x + \Theta_2 x^2 + \ldots + \Theta_q x^q \tag{3}$$

are matrix-valued polynomials with dimensions of p (order of regression) and q (order of moving average). Z_t denotes a multivariate "white noise" process.

There is one major drawback to this approach in our problem setting. The model treats all historical input values as stochastic variables. However, the product price does not vary *stochastically*, its value is preset by the vendor. Economic models assume a causal dependency between the price of a product and its sales quantities (see Arnold [15], chap. 17). Variations in consumer demand are caused by various factors like price, promotions, etc, Vorst [16] . This causal dependency is not modeled by VAR methods. This is an issue for the multivariate model.

Another complication that can arise in time series prediction is a noisy periodicity resulting, e.g., from low volume sales. The low sales quantity introduces a kind of random pattern that makes it hard to find even a known periodicity. In the sample data we used for our work, the overall sales history shows a clear 7-day periodicity (see Fig. 1) but not for most of the individual products. Therefore, we were looking for a method to calculate the eventually existing periodicity on product level. This kind of calculation is called periodicity mining and has received some attention from the research community lately.

Elfeky, Aref and Elmagarmid [17] introduce a periodicity mining algorithm that is based on symbols and a Fourier transformation inspired convolution. They defined two types of periodicity (segment and symbol periodicity) and described an convolution based algorithm for both of them. Our suggested algorithm uses the 'shifting' mechanic similar to Elfekys idea, but it differentiates on how similarity of a sequence of symbols is defined. The same authors also describe a method called "WARP" (WArping foR Periodicity) [18] in order to deal with noisy data. Thereby, their algorithm extends or shrinks the time axis at several locations of the time series to remove noise. Rasheed and Alhajj [19] recently used suffix trees (build by Ukonen's linear algorithm) as an underlying data structure in order to detect periodicity in time series. Their iterative approach decorates their suffix tree in a way that highlights repeated occurrences of a sequence of symbols.

Another approach for periodicity mining worth-mentioning was brought up by Berberidis, Aref, Atallah, Vlahavas and Elmagarmid [20]. They create a set of candidate periods out of a given time series and then use the autocorrelation function as well as Fast Fourier Transformation (FFT) for calculating a confidence value for each candidate period. We are also using candidate periods in our approach, but then we use the well known Pearson Correlation Coefficient in order to assess, which of our candidate periods is most suitable.

[1]We use parentheses () for a stochastic process instead of braces {} because it is rather a sequence of stochastic variables than a set

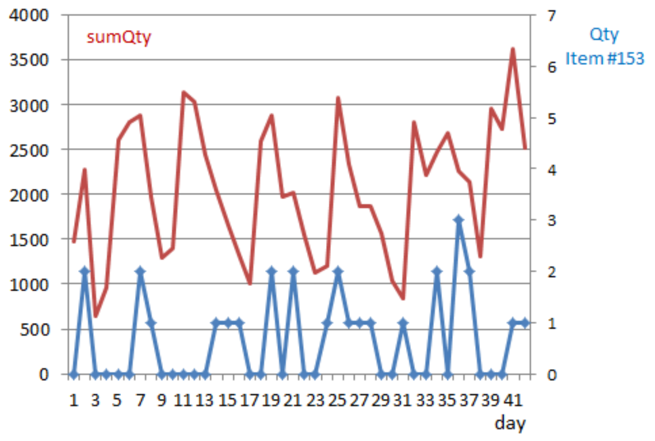


Fig. 1. Seven day periodicity for the overall sales data (DMC2012) and a typical low selling product (item # 153).

In the field of time series analysis, it is quite common to incorporate natural seasons or cycles into the prediction. Cyclic sales quantities are a typical behavior for short shelf-life products and are important for building a causal sales model. Doganis, Alexandridis, Patrinos and Sarimveis [21] investigated the sales quantity of fresh milk (a short shelf-life product) in Greece. They used a genetic algorithm applied to the sales quantities of the same weekday of last year. Our approach is only similar in that we take corresponding weekdays but it differs in how we analyze the weekly periodicity and correlate it with the sales prices.

To recapitulate, there are two general arguments against the multivariate VAR approach sketched above: Granger and Newbold [22] showed that simpler models often outperform forecasts based on complex multivariate models. And Lucas [13] criticized that the economic models are too static and that ”any change in policy will systematically alter the structure of the econometric model”. Applied to the sales forecast situation the variation of the price does not play a stochastic, but a *systematic*, i.e., a functional role.

Our idea is to filter the seasonality by a period-based ”folding” of the sales quantity, i.e., the aggregation of sales quantities for the same weekdays. This cancels the stochastic variation and accumulates the seasonal effect. Applying such a model improves the prediction coverage and accuracy for low volume data with a cyclic behavior.

III. Problem Description and Contribution

In Section I, we pointed out that the nature of the data and its sales profile play an important role for the time series analysis. In particular, the influence of price and periodicity are dominant factors as we will see in the following.

A. Formal Problem Description

The problem in terms of predicting time series consists of developing a parametrized time series model that is able to forecast future sales quantities depending on the given sales history and a price parameter. The solution of the stochastic Equation (1) is a multidimensional mapping

$$\begin{aligned} F: \quad (\mathbf{\Pi}, \mathbf{T}) &\longrightarrow (\mathbb{R}^+, \mathbb{N}_0) \\ (\pi, t) &\longmapsto (\hat{\pi}_r, \hat{x}_t) \end{aligned} \qquad (4)$$

where $\mathbf{\Pi}$ is the set of products and $\mathbf{T}$ are consecutive time intervals. A product $\pi \in \mathbf{\Pi}$ is described by its identification number π_i and its price π_r. The mapping F computes sales quantity $\hat{x}_t$ and price $\hat{\pi}_r$ for every product π and time interval t.

The bi-variate time series $(\hat{\pi}_r, \hat{x}_t)$ is a concrete realization of the stochastic process (X_t) of Equation (1). The mapping F has to be adjusted so that the process (X_t) explains best a given realization. This can be done by various estimator functions: least square error, Yule-Walker, maximum-likelihood, or Durbin-Levison algorithms. This is where our approach differs from the traditional because in real-life business the price is not a stochastic variable but is preset by the vendor. Instead of predicting the future price $\hat{\pi}_r$ we use the price as input parameter.

Having fixed the model in this way it is possible to transform the mapping F to the following form:

$$\begin{aligned} F_r: \quad (\mathbb{N}, \mathbb{R}^+, \mathbf{T}) &\longrightarrow \mathbb{N}_0 \\ (\pi_i, \pi_r, t) &\longmapsto \hat{x}_t \end{aligned} \qquad (5)$$

With this predictor $F_r(\pi_i, \pi_r, t)$ it is possible to forecast the sales quantities for future time periods $t > T$ (T is the present time) of a product $\pi \in \Pi$ using the future price π_r as input.

B. Contribution

By restricting our approach to model a linear trend, seasonality, and using historic and future prices as causal parameter leads to a predictor function that is easy to compute and explain. It yields higher accuracy for data with hidden periodicity and variable prices than the ARIMA model. The novelty of our contribution comprises:

- a model that has a causal explanation
- where the future price is a major input factor
- the overall periodicity is respected by individual items
- an algorithm for fast and automated periodicity mining

The prediction function can also be used for simulation to see how the price will influence the sales quantity. In addition to that, we introduce a new approach for periodicity mining, which is able to identify complex seasonal components within a time series. It is based on a simple form of data folding and comparisons of Pearson correlation coefficients.

IV. Data Profile

We used two types of data sets in this article. The first one was obtained from the DataMiningCup (DMC) in 2012. This real life data set was used to assess the prediction performance of our time series algorithm. It will be more closely described in the following subsection. During the development of our periodicity mining method, we also created an artificial data set in order to vary different aspects of a time series, such as

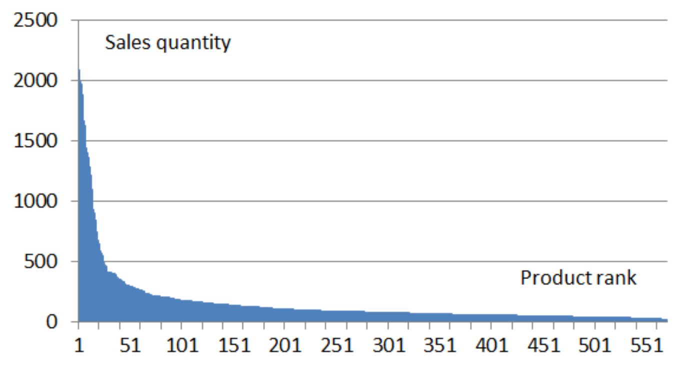


Fig. 2. Sales quantity ranking of sample data (DMC2012).

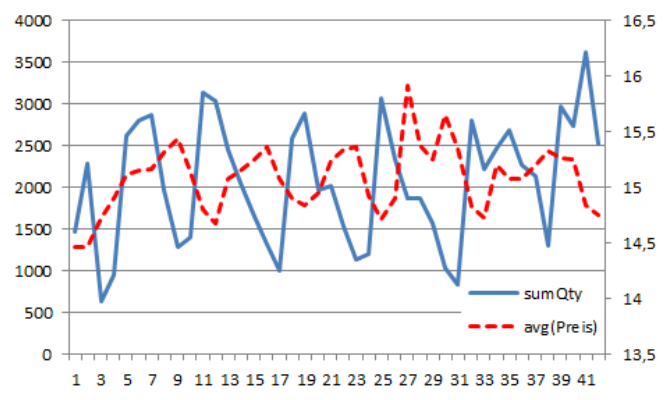


Fig. 3. Sales quantity and average price time series of sample data (DMC2012).

noise, time series length or season length (length of period). The method to create these data is detailed in subsection IV-2.

1) DMC data: The 570 products of the sample data realized a total of 86641 units sold. The average price for all products of a day ranged between 14.46 and 15.92 over the given time series, which included a total of 42 days. The maximum price variability of a single product is $\pm 48\%$, but on average the price varies only by $\pm 9\%$. However, for high selling products (> 500 units) the variability stands at $\pm 15\%$.

The total sales quantity per product ranged from 17 to 2083 over the 6 weeks. Broken down to the day level the product price ranged from 0.24 (cheapest product) to 152.92 (most expensive product) and the sales quantities between 0 and 193. The sample had average sales per product of 152 units with a standard deviation of 257, which indicates a high sales variability of the items.

This conjecture is confirmed by the product sales ranking that roughly follows a shifted hyperbolic distribution (see Fig. 2), which supports the assumption that low volume sales contribute significantly to the overall sales and may not be neglected. From the total of 570 products, the majority (506 products) sell less than 250 units in total, but contribute with approximately the same quantity sold (43991 units) as the 64 high selling products.

The low volume sales (sum of sales < 250) showed a strong positive trend ($\approx 40\%$ increase over 42 days) whereas the high volume sales (sum of sales ≥ 250) had a more stationary behavior. In the sample data are more than 100 products that sell less than six units a day. Nearly all of them sell none at half of the time.

The above properties require an adequate forecasting algorithm, which is able to handle low volume sales with high variability and which is also able to adapt to some price variability.

2) Artificial data set: The creation of artificial time series enabled us to accurately modify different aspects of a time series. This allowed us to investigate on how our periodicity mining algorithm reacts on changes of different parameters. Our goal was to create time series that show rather complex, periodic behavior with some added noise of various intensity.

The time series x_t of length n consist of a constant value c, which is modified by a repeating seasonal components S, which are (again) modified by a randomized noise parameter r as well as an noise intensifying factor f. Seasonal components consist of values $s \in S$ and have the length of m. The length n of a time series is the length m of one season multiplied by the number of seasons. The seasonal component repeats l times [see Equation (8)] along the time series to be created.

$$T = \{t_i | \forall i = 1, \dots, n\} \quad (6)$$
$$S = \{s_j | \forall j = 1, \dots, m\} \quad (7)$$
$$n = l * m \quad (8)$$
$$i \bmod m = j \quad (9)$$

A value of the constructed time series is calculated as the sum of constant c and the value of the current season value multiplied by the noise factor r and intensifying factor f:

$$x_{ti} = c + (s_j * r * f) \quad (10)$$

The subsequent value in that time series x_{ti+1} is calculated by shifting the seasonal component one position:

$$x_{ti+1} = c + (s_{j+1} * r * f) \quad (11)$$

This allowed us to create constant time series with an additive period that is disguised by a strong multiplicative interference factor. We created a total of 12 time series for this work with the following specifications:

- season length m: varied between 7 and 33 days
- time series length: 384 days
- basic constant c: was set to 100
- randomized noise parameter r: ranged between 0< r <1
- noise intensifying factor f: varied between 1, 5, and 10

An excerpt of one of the created time series can be seen in Fig. 4. The name of the shown time series is 29_days_1_n, this indicates that a 29 days long periodic component was used and also a noise intensifying factor of 1. The corresponding periodic component, which was used in this example, can be seen in Fig. 5.

Our intention was to create rather difficult seasonal components. We used several 'hand made' components. In addition

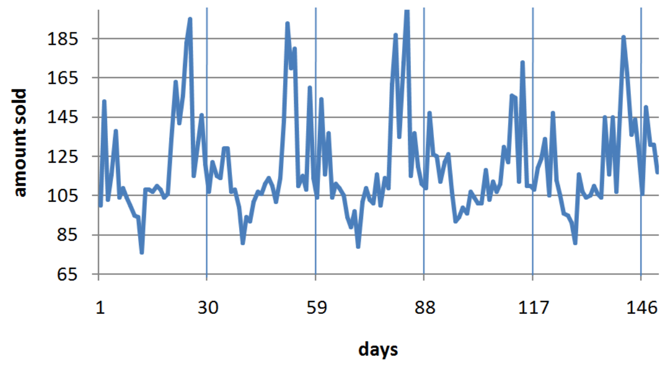


Fig. 4. This example time series was created by using a noise factor of 1 and the seasonal component from Fig. 5 with a length of 29. The vertical blue lines indicate the end of one seasonal component.

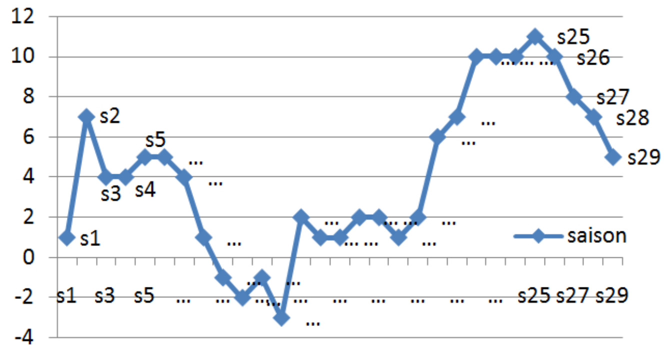


Fig. 5. This example seasonal profile has a length of 29 and was created using a noise factor of 1.

to that we used the following sinus Formula for creating some of the seasonal profiles:

$$\sin(\frac{2\pi}{m} * t) + \frac{1}{3} * \sin(\frac{2\pi * 3}{m} * t) \tag{12}$$

The parameter in above Equation is the base frequency, which determines the period length after the pattern is repeated. Variable t represents the time. This Formula creates time series with a base frequency as well as its third harmonic. An example of a sinus based time series is shown in Fig 6.

V. The Parametrized Time Series Model

This section describes the process from our initial analysis of the data for the DMC 2012, to the justification of our suggested time series model. For the causal predictor function F_r we needed to identify and quantify all influencing factors. Therefore, we firstly analyzed correlations of the attributes quantity, price and time of all products given in the DMC 2012 task. We used the standard Pearson Correlation [23] as a measure to determine the linear dependence between two time series. It is widely used and can range between -1 and $+1$. The following subsections will present the relations that have been analyzed.

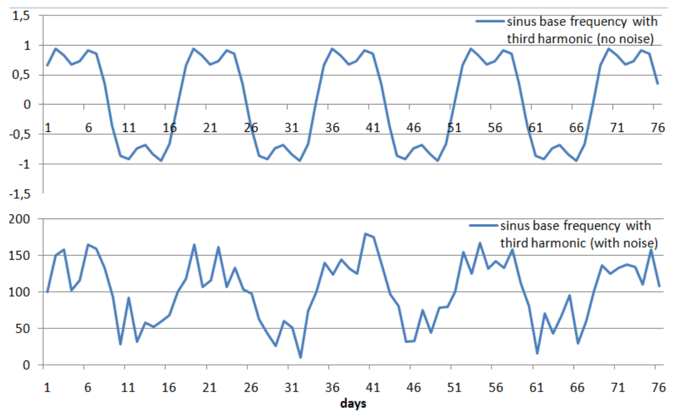


Fig. 6. Upper part: Example of a Sinus based time series with a period length of 17 days and no noise added. Lower part: shows the same time series, which was altered by a noise intensifying factor of 10.

A. Price-Sales Correlation

The main conjecture was that the price has a causal influence on the quantity. This is justified by the price elasticity of demand theory by Alfred Marshall [24]. As the correlation coefficients of all 570 products ranged from -0.6515 to $+0.3471$, we expected that the products with strong correlation exhibit a better prediction accuracy. Surprisingly, this seemed not to be the case.

A systematic analysis with three synthetic time series lead to an explanation. The first series had a growing price trend, the second and third had a cyclic price development where one product responded immediately and the other responded with a delay. ARIMA did recognize the price trend but forecasted a constant quantity instead of a decreasing one. This was the result of the low integer sales numbers that produced a monotone decreasing step function. Our approach managed to forecast the right quantities as long as a matching price was found in the history.

Surprisingly ARIMA could not deal well with the systematic cyclic price development and a detailed analysis showed that the step function of the price (which was kept constant for two days) was the reason. Fig. 7 shows the result of the ARIMA compared to our F_r algorithm (see Equation (5)). The reduced extrema produced by our algorithm results from the delay in the response to the price change. Without lag, no damping of extrema occurs in F_r.

B. Price Similarity

We also analyzed correlations between the price development of different products. The assumption was to find product bundles that are linked together via their price development. For the analysis the prices were normalized first in order to be able to easily compare the different price levels. Several bunches of products were linked together via their prices. But the corresponding sales figures of these products were not related. This is why we ignored the possible cross price influence from other products for the forecast.

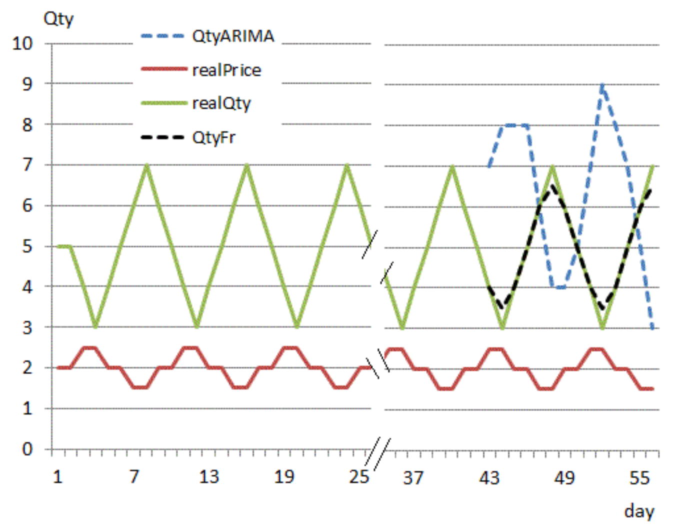


Fig. 7. Forecast of synthetic time series with delayed price-sales dependency.

C. Sales Periodicity

One of the most interesting properties of the DMC 2012 data was the periodicity of the total sales curve. It showed a clear 7 day period (Fig. 1). This period was not directly observable in most sales time series of individual products. Also the Pearson Correlation between the sum curve and the single products was too low to draw any conclusions. Nevertheless, since the total sales curve consists of all products, there must be a hidden periodicity within the individual products.

A systematic spectral analysis discovered not only the dominant weekly patterns but weaker 4, 5 and 14 days patterns. Since these periods differ on item level, there is the need for an automated periodicity mining method. For our work we developed JaPerCalc (Java Periodicity Calculator), which is detailed in Section VI.

D. The Parametrized Predictor Function

Putting all correlation observations together the result is a function F_r whose pseudo code is shown in Fig. 9. As input, it takes the price π_r of a product π at the prediction day t, the periodicity m and a price range δ. The upper and lower price limits are set to $\pm\delta$ percent. Using the periodicity from the previous analysis the algorithm looks for prices $\pi_r(w)$ that occur on days $w = t$ modulo $period$. For example, if the prediction day is 43 and JaPerCalc returns a suggested periodicity of 7 days, only the information from the days 36, 29, 22, 15, 8 and 1 will be considered (see Fig. 8). If the price on such a day is outside of the upper or the lower limit (day 1 in our example), the sales quantity is ignored. If the price is within the bounds, the corresponding quantity is selected. After all matching quantities have been selected, the forecast quantity is computed as linear trend of these quantities.

If all prices are outside of the upper and lower limit, no forecast is produced. The procedure may be repeated with enlarged upper and lower limits if needed. The F_r algorithm defines a simple forecasting model that takes into account the sales trend, the periodicity, and the price influence to predict sales quantities.

VI. Periodicity Calculation

In the normal case, the periodicity of a time series is unknown. However, it is beneficial for most time series algorithms to use this kind of information. The normally suggested standard algorithms [17] - [20] used to calculate this information are based on Fourier Transformation (FT), which is dependent on long time series histories. In order to automate and reduce complexity of the periodicity mining process, we developed our own approach, which will be detailed in the following subsections.

A. General Idea

JaPerCalc is an iterative approach in which a range of candidate periods (e.g., from 5 days period as minimum to 40 days period maximum) are tested. These upper and lower boundaries are the only input that needs to be defined prior by the user. For our hypothetical example, lets assume that we start with candidate period of 5 days. The first step is to fold the data accordingly to the given candidate period (in our case 5 days). This means that we sum up the sold quantity on every 5^{th} day (e.g., day 1, 6, 11, 16 etc.). This is repeated for each consecutive day until our candidate array is of length 5.

The next step is to take this folded candidate array and evaluate it against the given time series using the Pearson Correlation. In terms of our hypothetical example, we compare the folded 5 day candidate array with the days 1-5, 6-11, 11-16 etc. Each comparison results in a Pearson Correlation value r. Thereby, r is given as the mean of the products of the standard scores as it can be seen in Equation (13).

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad (13)$$

The average from all of these Person Correlation values is then associated with the candidate period of 5 days.

The process is repeated for all periods within the given boundaries. The finally suggested period is the one with the highest average Pearson Correlation Coefficient.

B. Formula Description

As mentioned before JaPerCalc takes a lower and upper period boundary (l, u) as well as a time series under examination as user input. All possible periods between the mentioned boundaries are referred as candidate periods. The algorithm starts by selecting the first suggested l period and use it to fold (i.e., sum up) every modulo c_j days of the given time series x_t (j is the length of candidate period c_j, see also Fig. 11). The resulting candidate period is held against all parts b of time series x_t of corresponding length. This process is also visualized in Fig. 10.

The algorithm calculates the Pearson correlation coefficient r for each pair of c_j and b. The average of these correlations is associated with the length j of the current candidate period

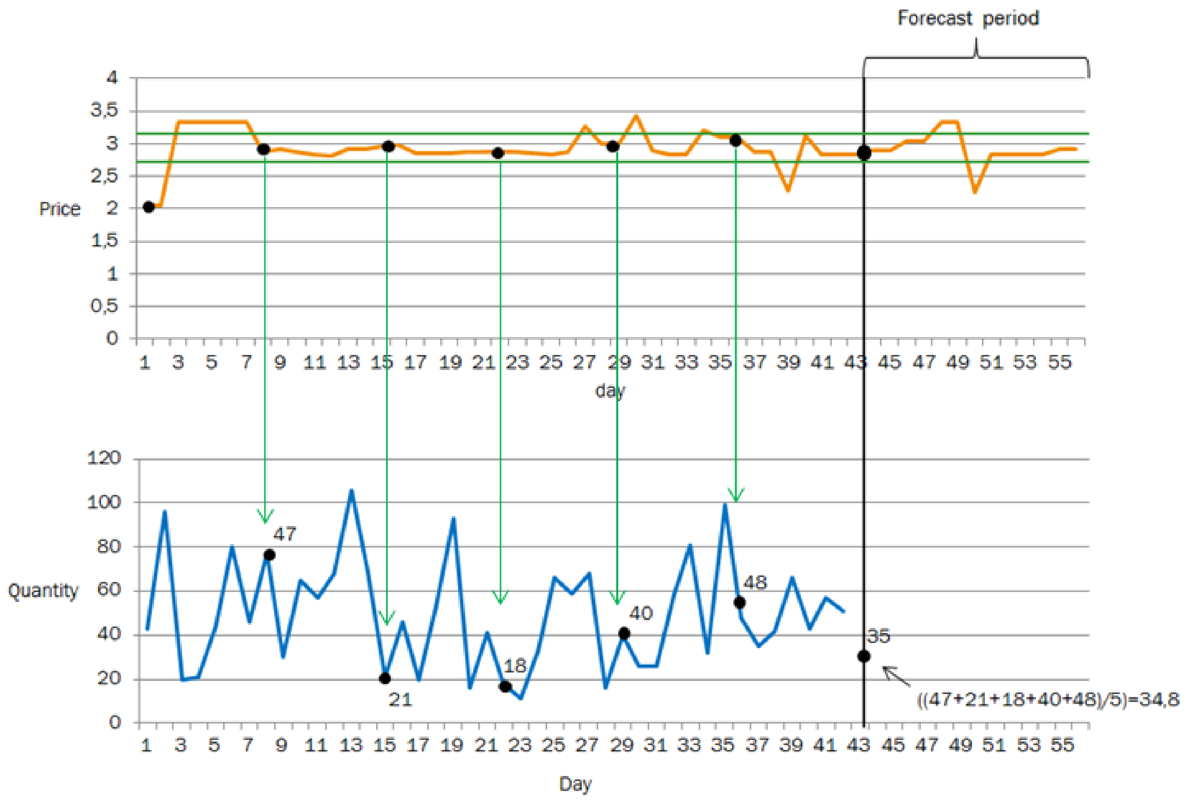


Fig. 8. Illustration of prediction concept using trend, a 7 days periodicity, and a parametrized price.

```
Input: $t > n$ // prediction day
    $\pi_r$ (input) // price at day t
    $\delta$ (input) // price range (e.g. $\pm$ 10%)
    $period$ (input) // period suggested by JaPerCalc
Def: $u, l$ // upper & lower price limit
    QtyList // list of sales quantities
    $w$ // = t - n * period ($n \geq 0$)
    $\hat{x_t}$ // predicted quantity at day t
for each $\pi \in \Pi$ {
    $u := \pi_r(1 + \delta/100);\ l := \pi_r(1 - \delta/100)$
    $w := t - period$
    while $(w \geq 1)$ {
      if $(\pi_r(w) < u)\,\&\,(\pi_r(w) > l)$
        QtyList.add($x_w$)
      $w := w - period$
    }
    if (QtyList $\neq \emptyset$)
      $\hat{x_t} := trend$(QtyList)
      return $\hat{x_t}$
    else
      return nil
}
```

Fig. 9. Parametrized Sales Prediction Algorithm F_r in conjunction with JaPerCalc.

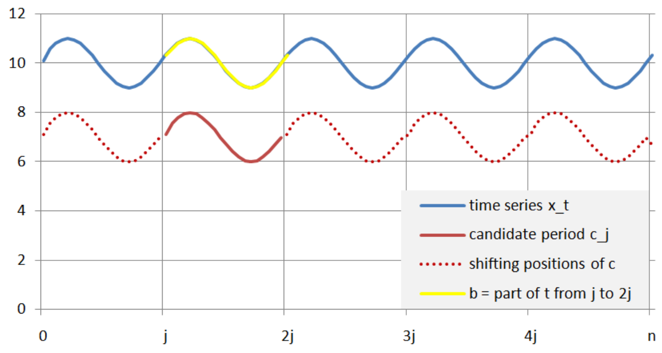


Fig. 10. Illustrative example of JaPerCalc comparing a candidate period c_j with all b parts of a time series x_t of length n.

and put into $PearsonResult$ array for later use. This process is repeated for each candidate period between the lower boundary l and the upper boundary u. The length of c_j is thereby incremented by one for each round. After retrieving the average Pearson correlation coefficient for each candidate period, the algorithm simply returns the j for the candidate component with the highest Pearson correlation coefficient. The complete pseudo code can also be seen in Fig. 11.

VII. Technical Framework and Infrastructure

This section covers some technical details about execution and implementation of the algorithms mentioned in this article.

```
Input: $(x_{ti}), i = 1, 2, ..., n$ // time series length n
    $u$ (input) // upper bound candidate period
    $l$ (input) // lower bound candidate period
Def: PearsonList // Set of Pearson corr coefficients
    PearsonResult // stores average Pearson results
    $r$ // Pearson correlation coefficient
    $c_j$ // candidate period array of length $j$
    $b$ // part of time series $x_t$, from day $e$ to day $r$
    $y$ // integer to iterate through $c_j$
    $x$ // integer to shift $b$ through $x_t$
    $iter := 0$
while $l + iter \leq u\{$
    $j := l + iter$
    $y = 0$
    for $(y < j)$ {
      //create candidate period $c_j$ of length $j$
      $c_y = sum_{i+(y \equiv mod(j))}(x_{ti})$
      $y$++
      }
    repeat $(\lfloor n/j \rfloor)$ times {
      //compare component $c_j$ with all parts $b$
      //of time series $(x_t)$
      $x = 0$
      $b$ = $(t_i | \forall i : x * j \leq i < x * j + j)$
      $r := PearsonCorr(c_j, b)$
      PearsonList.add($r$, $j$)
      $x$++
      }
    PearsonResult.add(PearsonList.getAvg($r$), $j$))
    $iter$++
    }
if (PearsonList $\neq \emptyset$)
return $j$ from $PearsonResult$
        where $r = PearsonResult.getMax(r)$
else
return nil
```

Fig. 11. Pseudo-code for JaPerCalc. Returns a suggested period for a given timeline.

A. ARIMA Model Execution

The Microsoft Visual Studio 2008 and Microsoft SQL Server 2008 were used to apply the ARIMA model on the two given data sets (DMC and artificial data). In order to run the ARIMA mining models for both data sets, a OLAP cube was build. It consists of the dimensions *price*, *product* and *time*. In the corresponding time series mining model we used *itemId* and *day* as key attributes and the *price* attribute as input. The *quantity* was set as predictable attribute. For the DMC data we used mostly the default parameters, apart from the the minimum series value and the periodicity hint. The following table shows all model parameters used:

AUTO_DETECT_PERIODICITY	0.6
FORECAST_METHOD	ARIMA
HISTORIC_MODEL_COUNT	1
HISTORIC_MODEL_GAP	10
INSTABILITY_SENSITIVITY	1.0
MAXIMUM_SERIES_VALUE	$+1E308$
MINIMUM_SERIES_VALUE	0
MISSING_VALUE_SUBSTITUTION	None
PERIODICITY_HINT	7
PREDICTION_SMOOTHING	0.5

The specification for the artificial data is equal to the one shown above, except for the PERIODICTIY_HINT that was left blank.

B. Implementation of Time Series Model F_r

Our suggested approach was implemented in Java. We used Eclipse (Version: Indigo Service Release 1) with Java Platform Standard Edition 6.0 (JRE6). The data was stored in a MySQL database on an Apache web server (2.2.21). During execution time the data is queried from the database, the model parameters computed and the forecast results are instantly stored in the corresponding result table in the database. The model was developed using the standard java.sql.* package, which was used to interface with the database and for SQLException handling.

C. Implementation of Periodicity Calculation JaPerCalc

As the name indicates, JaPerCalc was also implemented in Java. We used the two freeware libraries java.util.* and org.apache.commons.math3.stat.* in order to compute the Pearson correlation. We implemented two version of it. The first one outputs the suggested period as well as statistics about the average Pearson correlations of the compared components. It was used for pitching JaPerCalc against Fourier Transformation, see also Section VIII-B. The second version simply returns the suggested period and was used within the F_r algorithm for periodicity detection.

VIII. Experiments and Results

There have been three different sets of experiments, that we carried out. The first set used data from the DataMiningCup 2012. We applied our suggested F_r algorithm in order to predict sales quantities for a given price. We compared its results with the predictions from an ARIMA implementation described in Section VII-A. A discussion about the results can be found in the following Section VIII-A. During the time of the DataMiningCup 2012 we assumed a hidden periodicity within the data on a item level [1]. However, there were some exceptions from our assumption, which raised the need for an automated periodicity detection.

In order to meet this need, we were looking for periodicity detection methods. Our experiments with Fourier transformation led to rather unsatisfactory results and this was the starting point for the development of JaPerCalc. We compare both approaches in Section VIII-B.

The third set of experiments includes both, the time series prediction as well as the periodicity detection problem. We

TABLE II
COMPARISON OF ARIMA PREDICTION ERROR WITH F_r ALGORITHM ON DMC DATA

Class	ARIMA	F_r	improvement
All products	30512	22093	26.7%
quantity < 500	24338	17248	29.1%
quantity ≥ 500	6174	4845	21.5%
(quantity = 0) in $< 1/3$ time	19178	15942	16.9%
(quantity = 0) in $\geq 1/3$ time	11334	6151	45.7%
avg(π_r) < 20	26756	18711	30.1%
avg(π_r) ≥ 20	3756	3382	10.0%
top 100 items	15805	6131	61.2%
least 470 items	16167	15962	1.2%

TABLE III
COMPARISON OF PERIODICITY SUGGESTED BY FOURIER TRANSFORMATION AND F_r ON ARTIFICIAL DATA

time series name	FFT	F_r	real
16_days_1	16	16	16
17_days_10_n_sin	28	17	17
17_days_1_n_sin	17	17	17
17_days_5_n_sin	17	17	17
20_days_1_n	20	20	20
29_days_1_n	28	29	29
33_days_10_n	32	33	33
33_days_1_n	32	33	33
33_days_5_n	32	33	33
7_days_10_n	7	35	7
7_days_1_n	7	35	7
7_days_5_n	7	42	7

used artificial data for these experiments and compared to the results of ARIMA. We will discuss the results in Section VIII-C.

A. Comparison of Results with ARIMA

The absolute prediction error was measured as $|realQty - predictQty|$. The F_r algorithms benefited from two input parameters: the hidden periodicity that was calculated in a previous step and the predefined future price (see also [1] for more details). The hidden sales periodicity contributed for an improvement of about 20%. The overall forecast was improved by 26.7%. The price influence was less dominant than expected, but was determinant for a cluster of 26 products. Cluster characteristics:

- correlation < -0.25
- relative standard error < 0.25
- sales quantity > 160
- price variation $(max(\pi_r) - min(\pi_r)) > 4$

In total, F_r could forecast this cluster 36.4% better than ARIMA. Table II shows the prediction error points of both ARIMA and F_r on certain product clusters. These were grouped based on attributes which we guessed to have an impact on the prediction performance (e.g., the top 100 vs the least 470 items). The total error of all 570 products was 30152 for the ARIMA and 22093 for F_r. This is an improvement of 26.7% compared to ARIMA. For further analysis we clustered the products into disjoint sets according to different criteria. This allowed us to find the strengths and weaknesses of F_r in terms of total sales quantity, sales sparsity, and price.

B. Periodicity Calculation

This subsection will show some results from our experiments in which we compare Fourier Transformation with our suggested methods for periodicity mining on our artificial data set. Table III shows all results.

The FFT detected 8 out of 12 periods correctly. It seems like FFT performs better on high frequency periods with a shorter period length (e.g., 7, 16, 17). However, for longer periods (e.g., 33, 29), FFT slightly underestimated the periodicity length.

F_r was able to detect 9 out of 12 periods correctly. Please note that in all of the three wrong suggested cases, a multiple

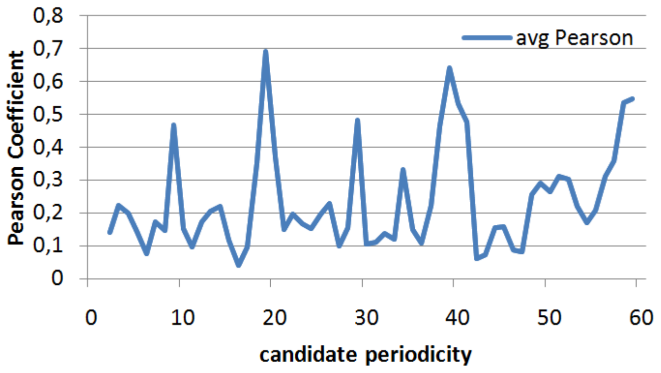


Fig. 12. Average Pearson Correlations for several candidate periods in time series 20_days_1_n. Please note that also multiple and partial candidate components are highlighted (i.e., the correct periodicity of 20 is highlighted, but also partial components such as 10, 30 or multiple components such as 40 or 60.)

of the correct period length (harmonic) was suggested. Reason for this lies within the intrinsic mechanic of JaPerCalc. If a periodicity of a certain length n is repeating within a time series, there is also a multiple of the inherent periodicity with length $2n$ ($4n$, $6n$ and so forth) visible. Same goes for partial periodicity of length $1.5n$ or $2.5n$. JaPerCalc is able to highlight all of these periodicity lengths. Fig. 12 shows this for the example of time series 20_days_1_n. In case of the wrong suggested periodicity from time series 7_days_*_n it is very likely that noise caused a multiple component of the inherent periodicity length to have a slightly higher average Pearson correlation coefficient than the inherent (given) periodicity. This can occur in following scenario: if a inherent period has a significant peak on, lets say the 7^{th} day, this periodicity is disguised if noise randomly causes every second of these peaks to diminish. Result is a higher average Pearson correlation coefficient for a period of length 14 (rather than the correct 7 days). How this affected the predictions made by F_r can be seen in the following section.

TABLE IV
COMPARISON OF ARIMA PREDICTION ERROR WITH F_r ALGORITHM ON ARTIFICIAL DATA

time series name	ARIMA	F_r + JaPerCalc	improvement
16_days_1	797	240	69.89 %
17_days_10_n_sin	10443	3400	67.44 %
17_days_1_n_sin	1048	304	70.99 %
17_days_5_n_sin	5165	1627	68.50 %
20_days_1_n	3002	1422	52.63 %
29_days_1_n	5006	2139	57.27 %
33_days_10_n	7883	3234	58.98 %
33_days_1_n	862	330	61.72 %
33_days_5_n	3776	1471	61.04 %
7_days_10_n	11307	9610	15.01 %
7_days_1_n	1154	937	18.80 %
7_days_5_n	5942	4576	22.99 %

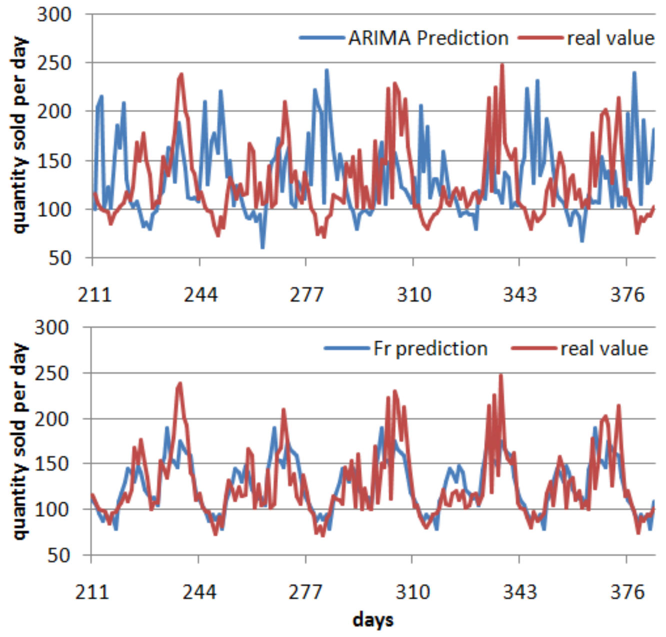


Fig. 13. prediction results for ARIMA (upper part) and prediction results for F_r (lower part).

C. Time Series Prediction with automated Periodicity Detection

As mentioned before we used artificial data described in IV-2. We once again calculated the absolute prediction error as $|realQty - predictQty|$. Table IV shows the results for all generated time series.

JaPerCalc was able to find the correct period for 9 out of the given 12 time series. The predictions for both compared methods for 33_days_10_n are visualized in Fig. 13. The period length prediction for 7_days_*_n was incorrectly predicted by JaPerCalc (as described in previous section). However, since a multiple of correct period was suggested, F_r is still able to outperform ARIMA by 15.01% to 22.99%. If the suggestions from JaPerCalc is correct, the prediction improvement from F_r compared to ARIMA rises to averaged 63.13%.

As it can be seen ARIMA is not able to compensate the noise within the time series and gets biased. F_r on the

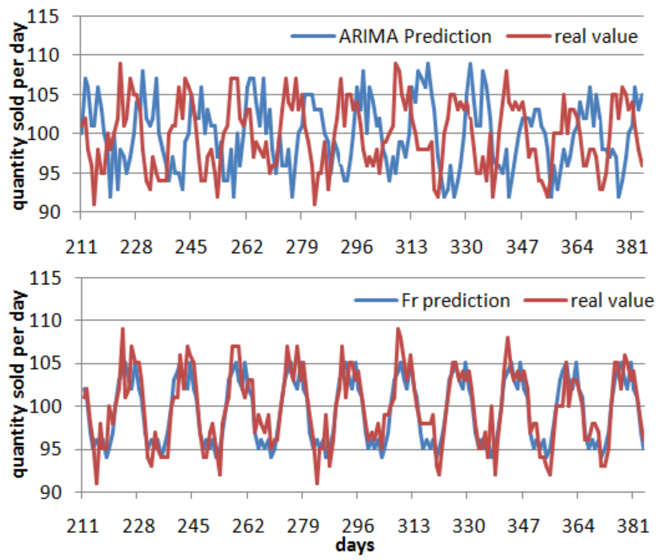


Fig. 14. prediction results for ARIMA (upper part) and prediction results for F_r (lower part). We used a sinus function to create this time series.

other, hand is able to average the given noise by following the candidate period, which was detected by JaPerCalc. The interaction of two rather simple algorithms is able to deliver impressive predictions results. Similar behavior can be seen for the time series created by the sinus function described in Equation (12). An example can be seen in Fig. 14.

As the name of the time series 17_days_1_noise indicates, there was only a noise factor of 1 used. Although a rather simple period component and low noise was used, ARIMA gets distracted and produced poor prediction results.

IX. CONCLUSION AND FUTURE WORK

This article covered a long development. We started with the DMC 2012 and its challenging data set. The broad range of products with its hidden periodicity made the analysis difficult. The low volume sales further complicated the analysis of the influence of the price on the sales quantities. The conclusions drawn from the above results can be summarized in the following three statements:

1) Data profiling and periodicity mining is crucial for choosing the best time series model
2) Low sales volume can hide a cyclic sales behavior and the price should be treated as input parameter
3) Simple models for sales forecasting based on causal parameters can outperform some sophisticated stochastic models.

This lead to the development of the rather simple F_r algorithm. If it is applied on other data, in which the periodicity is not known, it is likely to produce disappointing results. Reason for this is its dependency on the input period (which was set to 7 in case of the DMC 2012 data).

JaPerCalc was created in order to overcome that weakness. It is also a rather simple and fast algorithm that is able to detect periodicity in a given time series. The Pearson correlation coefficient allows to detect periodicity of any form or shape.

The combination of both simple algorithms is able to significantly outperform standard methods such as the ARIMA, which was shown in Section VIII. JaPerCalc as well as F_r algorithm can be used with incomplete time series. This is particularly useful for real-time analysis used in recommendation systems. The simplicity and low computational effort for both algorithms makes them ideal for people who want to delve into the field of time series prediction.

In terms of future work for F_r, a spectral analysis on an individual product level could further improve the prediction accuracy. For products with a strong monotone price development, our approach to look for similar prices is not well suited. The price trend should be computed instead. There is also the option to adjust the price ranges by the variance of the so far known time series. Products with a high variance could be allowed to have a broader prince range then product with a lower variance. JaPerCalc could also be adapted in a way to automatically recognize and adjust eventually existing trends. This would help to improve periodicity recognition for monotonous increasing time series. Another direction of development could be to use confidence values to help JaPerCalc to find the base frequency of a time series (i.e., shortest suitable period length). This could help to prevent incorrect suggestion of multiples of the correct period length as described in Section VIII-B.

References

[1] M. Schaidnagel, ”Sales Prediction with Parametrized Time Series Analysis”, DBKDA 2013, The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications, Seville, Spain - January 27^{th} - February 1^{st}, 2013

[2] A.-W. Scheer, *Sales Forecast*, Springer Verlag, Berlin, 1983

[3] M. Hüttner, *Market and Sales Forecast*, Kohlhammer, Stuttgart, 1982

[4] Y. Lan and D. Deagu, *A New Approach and Its Applications for Time Series Analysis and Prediction Based on Moving Average of n^{th}-Order Difference*, in: D. E. Holmes and L. C. Jain (Eds.), *Data Mining: Foundations and Intelligent Paradigms*, Vol 2: Statistical, Bayesian, Time Series and other Theoretical Aspects, pp. 157 - 182, Springer Berlin Heidelberg, 2012

[5] K. Neusser, *Time Series Analysis for Economic Science*, B. G. Teubner Verlag, Wiesbaden, 2006

[6] A. J. Izenman, *Modern Multivariate Statistical Techniques - Regression, Classification, and Manifold Learning*, Springer Science + Business Media, New York, 2008

[7] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*, corr. repr., Springer Verlag, Berlin Heidelberg, 2007

[8] S. Oches, ”Top 50 Sorted by Total Units”, Special Report of QSR Magazine, Journalistic Inc., August 2011, [Online] http://www.qsrmagazine.com/reports/top-50-sorted-total-units, last access: 14.12.2013

[9] Economist Intelligence Unit, ”Denmark: Market Indicators and Forecasts”, [Online] http://datamarket.com/data/set/1wmo/ (indicators: Private consumption, Consumer goods), last access: 30.08.2012

[10] J. Griese, *Adaptive Verfahren im betrieblichen Entscheidungsprozess*, Physica Verlag, Würzburg - Wien, 1972

[11] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 1^{st} rev. ed., Holden Day,Oakland, San Francisco, 1976

[12] C. A. Sims, ”Macroeconomics and Reality”, in: Econometrica, Vol. 48, No. 1, pp. 1 - 48, 1980

[13] R. E. Lucas, ”Econometric policy evaluation: A critique”, In: K. Brunner and A. H. Meltzer (Eds), *The Phillips Curve and Labor Markets*, Vol. 1, Carnegie-Rochester Conference Series on Public Policy, pp. 19 - 46, Amsterdam, North-Holland, 1976

[14] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2^{nd} Edition, Springer Science + Business Media, New York, 2006

[15] R. A. Arnold, *Economics*, 9^{th} ed., South-Western College Publ., 2008

[16] J. G. A. J. van der Vorst, A. J. M. Beulens, W. de Wit, P. van Beek, *Supply chain management in food chains: Improving performance by reducing uncertainty*, in: International Transactions in Operational Research, Vol. 5(6), pp 487 - 499, 1998

[17] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid *Periodicity Detection in Time Series Databases*, in: IEEE Transactions on Knowledge and Data Engineering, pp. 875 - 887 Vol. 17, No. 7, July 2005

[18] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid *WARP: Time Warping for Periodicity Detection*, in: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM05)

[19] F. Rasheed, R. Alhajj *Using Suffix Trees for Periodicity Detection in Time Series Databases*, in 4th International IEEE Conference ”Intelligent Systems”, 2008

[20] C. Berberidis, W. G. Aref, M. Atallah, I. Vlahavas, A. K. Elmagarmid *Multiple and Partial Periodicity Mining in Time Series Databases*, in F. van Harmelen (ed.): ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence, IOS Press, Amsterdam, 2002, pp.30-37

[21] P. Doganis, A. Alexandridis, R. Patrinos, and H. Sarimveis, *Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing*, Journal of Food Engineering 75, pp. 196 - 204, Elsevier Ltd., 2006, [Online] http://www.elsevier.com/locate/jfoodeng, last access: 30.08.2012

[22] C. W. J. Granger and P. Newbold, *Economic Forecasting: The Atheist's Viewpoint*, in: G.A. Renton (ed.), *Modelling the Economy*, pp. 131 - 148, Heinemann, London, 1975

[23] H. Rinne and K. Specht, *Time Series - statistical modelling, Estimation and Prediction*, Verlag Vahlen, Munich, 2002

[24] A. Marshall, *Principles of Economics*, 8^{th} ed., Cosimo Classics, 2009, first publ. 1890