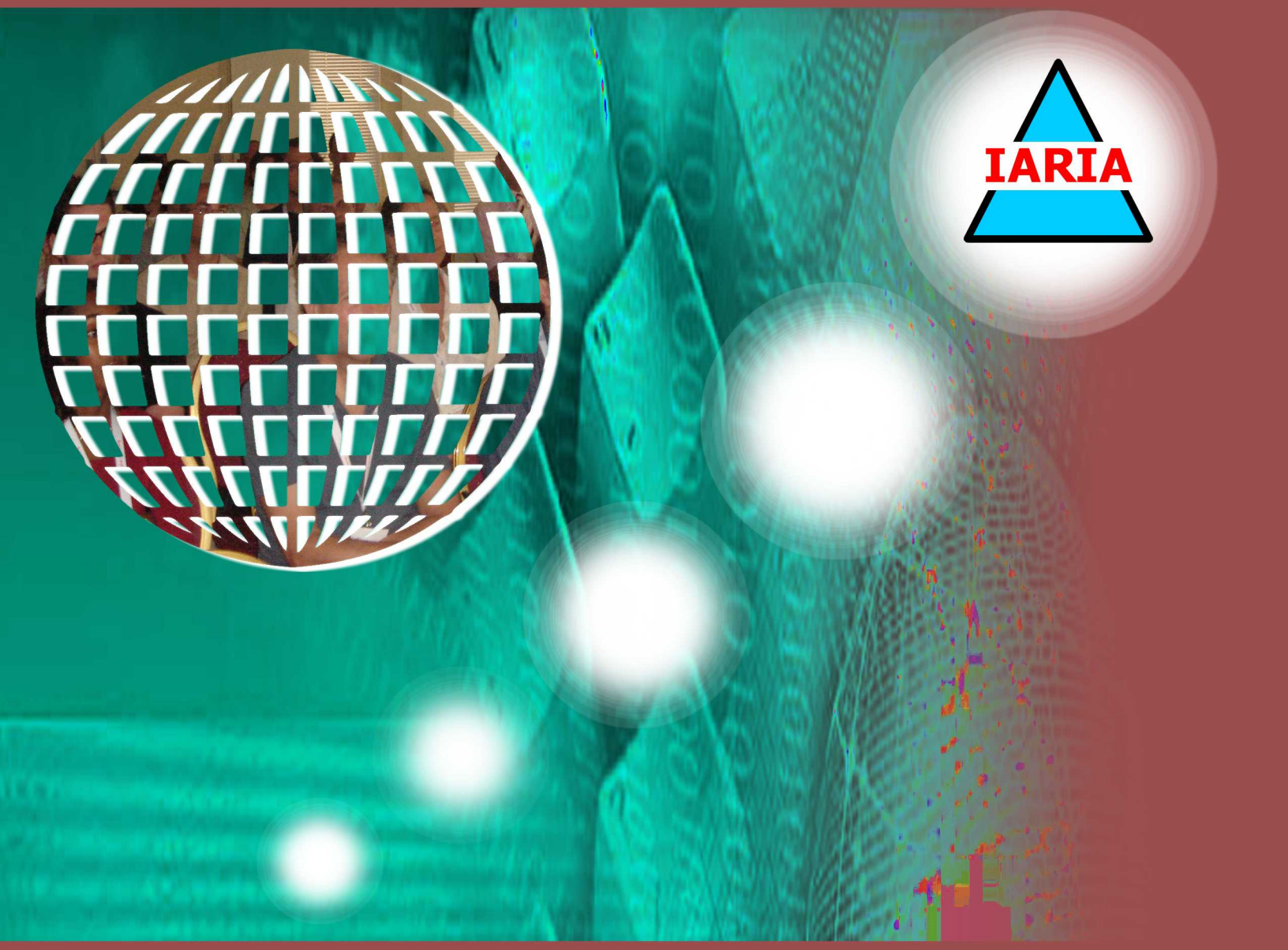
International Journal on Advances in Software



2023 vol. 16 nr. 1&2

The *International Journal on Advances in Software* is published by IARIA.
ISSN: 1942-2628
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Software
Volume 16, Number 1 & 2, 2023

Thierry Badard, Université Laval, Canada
Muneera Bano, International Islamic University - Islamabad, Pakistan
Fabian Barbato, Technology University ORT, Montevideo, Uruguay
Peter Baumann, Jacobs University Bremen / Rasdaman GmbH Bremen, Germany
Gabriele Bavota, University of Salerno, Italy
Grigorios N. Beligiannis, University of Western Greece, Greece
Noureddine Belkhatir, University of Grenoble, France
Jorge Bernardino, ISEC - Institute Polytechnic of Coimbra, Portugal
Rudolf Berrendorf, Bonn-Rhein-Sieg University of Applied Sciences - Sankt Augustin, Germany
Ateet Bhalla, Independent Consultant, India
Fernando Boronat Seguí, Universidad Politecnica de Valencia, Spain
Pierre Borne, Ecole Centrale de Lille, France
Farid Bourennani, University of Ontario Institute of Technology (UOIT), Canada
Narhimene Boustia, Saad Dahlab University - Blida, Algeria
Hongyu Pei Breivold, ABB Corporate Research, Sweden
Carsten Brockmann, Universität Potsdam, Germany
Antonio Bucchiarone, Fondazione Bruno Kessler, Italy
Georg Buchgeher, Software Competence Center Hagenberg GmbH, Austria
Dumitru Burdescu, University of Craiova, Romania
Martine Cadot, University of Nancy / LORIA, France
Isabel Candal-Vicente, Universidad Ana G. Méndez, Puerto Rico
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain
Jose Carlos Metrolho, Polytechnic Institute of Castelo Branco, Portugal
Alain Casali, Aix-Marseille University, France
Yaser Chaaban, Leibniz University of Hanover, Germany
Savvas A. Chatzichristofis, Democritus University of Thrace, Greece
Antonin Chazalet, Orange, France
Jiann-Liang Chen, National Dong Hwa University, China
Shiping Chen, CSIRO ICT Centre, Australia
Wen-Shiung Chen, National Chi Nan University, Taiwan
Zhe Chen, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China PR
Yoonsik Cheon, The University of Texas at El Paso, USA
Lau Cheuk Lung, INE/UFSC, Brazil
Robert Chew, Lien Centre for Social Innovation, Singapore
Andrew Connor, Auckland University of Technology, New Zealand
Rebeca Cortázar, University of Deusto, Spain
Noël Crespi, Institut Telecom, Telecom SudParis, France
Carlos E. Cuesta, Rey Juan Carlos University, Spain
Duilio Curcio, University of Calabria, Italy
Mirela Danubianu, "Stefan cel Mare" University of Suceava, Romania
Paulo Asterio de Castro Guerra, Tapijara Programação de Sistemas Ltda. - Lambari, Brazil
Cláudio de Souza Baptista, University of Campina Grande, Brazil
Maria del Pilar Angeles, Universidad Nacional Autonónoma de México, México
Rafael del Vado Vírseda, Universidad Complutense de Madrid, Spain
Giovanni Denaro, University of Milano-Bicocca, Italy

Ismail Hababeh, German Jordanian University, Jordan
Shahliza Abd Halim, Lecturer in Universiti Teknologi Malaysia, Malaysia
Herman Hartmann, University of Groningen, The Netherlands
Jameleddine Hassine, King Fahd University of Petroleum & Mineral (KFUPM), Saudi Arabia
Tzung-Pei Hong, National University of Kaohsiung, Taiwan
Peizhao Hu, NICTA, Australia
Chih-Cheng Hung, Southern Polytechnic State University, USA
Edward Hung, Hong Kong Polytechnic University, Hong Kong
Noraini Ibrahim, Universiti Teknologi Malaysia, Malaysia
Anca Daniela Ionita, University "POLITEHNICA" of Bucharest, Romania
Chris Ireland, Open University, UK
Kyoko Iwasawa, Takushoku University - Tokyo, Japan
Mehrshid Javanbakht, Azad University - Tehran, Iran
Wassim Jaziri, ISIM Sfax, Tunisia
Dayang Norhayati Abang Jawawi, Universiti Teknologi Malaysia (UTM), Malaysia
Jinyuan Jia, Tongji University. Shanghai, China
Maria Joao Ferreira, Universidade Portucalense, Portugal
Ahmed Kamel, Concordia College, Moorhead, Minnesota, USA
Teemu Kanstrén, VTT Technical Research Centre of Finland, Finland
Nittaya Kerdprasop, Suranaree University of Technology, Thailand
Ayad ali Keshlaf, Newcastle University, UK
Nhien An Le Khac, University College Dublin, Ireland
Sadegh Kharazmi, RMIT University - Melbourne, Australia
Kyoung-Sook Kim, National Institute of Information and Communications Technology, Japan
Youngjae Kim, Oak Ridge National Laboratory, USA
Cornel Klein, Siemens AG, Germany
Alexander Knapp, University of Augsburg, Germany
Radek Koci, Brno University of Technology, Czech Republic
Christian Kop, University of Klagenfurt, Austria
Michal Krátký, VŠB - Technical University of Ostrava, Czech Republic
Narayanan Kulathuramaiyer, Universiti Malaysia Sarawak, Malaysia
Satoshi Kurihara, Osaka University, Japan
Eugenijus Kurilovas, Vilnius University, Lithuania
Alla Lake, Linfo Systems, LLC, USA
Fritz Laux, Reutlingen University, Germany
Luigi Lavazza, Università dell'Insubria, Italy
Fábio Luiz Leite Júnior, Universidade Estadual da Paraiba,Brazil
Alain Lelu, University of Franche-Comté / LORIA, France
Cynthia Y. Lester, Georgia Perimeter College, USA
Clement Leung, Hong Kong Baptist University, Hong Kong
Weidong Li, University of Connecticut, USA
Corrado Loglisci, University of Bari, Italy
Francesco Longo, University of Calabria, Italy
Sérgio F. Lopes, University of Minho, Portugal
Pericles Loucopoulos, Loughborough University, UK
Alen Lovrencic, University of Zagreb, Croatia

Qifeng Lu, MacroSys, LLC, USA
Xun Luo, Qualcomm Inc., USA
Stephane Maag, Telecom SudParis, France
Ricardo J. Machado, University of Minho, Portugal
Maryam Tayefeh Mahmoudi, Research Institute for ICT, Iran
Nicos Malevris, Athens University of Economics and Business, Greece
Herwig Mannaert, University of Antwerp, Belgium
José Manuel Molina López, Universidad Carlos III de Madrid, Spain
Francesco Marcelloni, University of Pisa, Italy
Eda Marchetti, Consiglio Nazionale delle Ricerche (CNR), Italy
Gerasimos Marketos, University of Piraeus, Greece
Abel Marrero, Bombardier Transportation, Germany
Adriana Martin, Universidad Nacional de la Patagonia Austral / Universidad Nacional del Comahue, Argentina
Goran Martinovic, J.J. Strossmayer University of Osijek, Croatia
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal
Stephan Mäs, Technical University of Dresden, Germany
Constandinos Mavromoustakis, University of Nicosia, Cyprus
Jose Merseguer, Universidad de Zaragoza, Spain
Seyedeh Leili Mirtaheri, Iran University of Science & Technology, Iran
Lars Moench, University of Hagen, Germany
Yasuhiko Morimoto, Hiroshima University, Japan
Antonio Navarro Martín, Universidad Complutense de Madrid, Spain
Filippo Neri, University of Naples, Italy
Muaz A. Niazi, Bahria University, Islamabad, Pakistan
Natalja Nikitina, KTH Royal Institute of Technology, Sweden
Roy Oberhauser, Aalen University, Germany
Pablo Oliveira Antonino, Fraunhofer IESE, Germany
Rocco Oliveto, University of Molise, Italy
Sascha Opletal, Universität Stuttgart, Germany
Flavio Oquendo, European University of Brittany/IRISA-UBS, France
Claus Pahl, Dublin City University, Ireland
Marcos Palacios, University of Oviedo, Spain
Constantin Paleologu, University Politehnica of Bucharest, Romania
Kai Pan, UNC Charlotte, USA
Yiannis Papadopoulos, University of Hull, UK
Andreas Papasalouros, University of the Aegean, Greece
Rodrigo Paredes, Universidad de Talca, Chile
Päivi Parviainen, VTT Technical Research Centre, Finland
João Pascoal Faria, Faculty of Engineering of University of Porto / INESC TEC, Portugal
Fabrizio Pastore, University of Milano - Bicocca, Italy
Kunal Patel, Ingenuity Systems, USA
Óscar Pereira, Instituto de Telecomunicacoes - University of Aveiro, Portugal
Willy Picard, Poznań University of Economics, Poland
Jose R. Pires Manso, University of Beira Interior, Portugal
Sören Pirk, Universität Konstanz, Germany
Meikel Poess, Oracle Corporation, USA

Thomas E. Potok, Oak Ridge National Laboratory, USA
Christian Prehofer, Fraunhofer-Einrichtung für Systeme der Kommunikationstechnik ESK, Germany
Ela Pustułka-Hunt, Bundesamt für Statistik, Neuchâtel, Switzerland
Mengyu Qiao, South Dakota School of Mines and Technology, USA
Kornelije Rabuzin, University of Zagreb, Croatia
J. Javier Rainer Granados, Universidad Politécnica de Madrid, Spain
Muthu Ramachandran, Leeds Metropolitan University, UK
Thurasamy Ramayah, Universiti Sains Malaysia, Malaysia
Prakash Ranganathan, University of North Dakota, USA
José Raúl Romero, University of Córdoba, Spain
Henrique Rebêlo, Federal University of Pernambuco, Brazil
Hassan Reza, UND Aerospace, USA
Elvinia Riccobene, Università degli Studi di Milano, Italy
Daniel Riesco, Universidad Nacional de San Luis, Argentina
Mathieu Roche, LIRMM / CNRS / Univ. Montpellier 2, France
José Rouillard, University of Lille, France
Siegfried Rouvrais, TELECOM Bretagne, France
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Djamel Sadok, Universidade Federal de Pernambuco, Brazil
Ismael Sanz, Universitat Jaume I, Spain
M. Saravanan, Ericsson India Pvt. Ltd -Tamil Nadu, India
Idrissa Sarr, University of Cheikh Anta Diop, Dakar, Senegal / University of Quebec, Canada
Patrizia Scandurra, University of Bergamo, Italy
Daniel Schall, Vienna University of Technology, Austria
Rainer Schmidt, Munich University of Applied Sciences, Germany
Sebastian Senge, TU Dortmund, Germany
Isabel Seruca, Universidade Portucalense - Porto, Portugal
Kewei Sha, Oklahoma City University, USA
Simeon Simoff, University of Western Sydney, Australia
Jacques Simonin, Institut Telecom / Telecom Bretagne, France
Cosmin Stoica Spahiu, University of Craiova, Romania
George Spanoudakis, City University London, UK
Cristian Stanciu, University Politehnica of Bucharest, Romania
Lena Strömbäck, SMHI, Sweden
Osamu Takaki, Japan Advanced Institute of Science and Technology, Japan
Antonio J. Tallón-Ballesteros, University of Seville, Spain
Wasif Tanveer, University of Engineering & Technology - Lahore, Pakistan
Ergin Tari, Istanbul Technical University, Turkey
Steffen Thiel, Furtwangen University of Applied Sciences, Germany
Jean-Claude Thill, Univ. of North Carolina at Charlotte, USA
Pierre Tiako, Langston University, USA
Božo Tomas, HT Mostar, Bosnia and Herzegovina
Davide Tosi, Università degli Studi dell'Insubria, Italy
Guglielmo Trentin, National Research Council, Italy
Dragos Truscan, Åbo Akademi University, Finland

International Journal on Advances in Software
Volume 16, Numbers 1 & 2, 2023

CONTENTS

A Formalism for Explaining Concepts through Examples based on a Source Code Abstraction

Mirco Schindler*, Christian Schindler† and Andreas Rausch‡
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal, Germany
* Email: mirco.schindler@tu-clausthal.de
† Email: christian.schindler@tu-clausthal.de
‡ Email: andreas.rausch@tu-clausthal.de

***Abstract*—Design and architecture patterns are proven domain-independent solution approaches for common problems occurring in the development of software systems. Correct implementation of the design pattern is essential to guarantee the problem-solving capabilities of patterns. As the developers need to perform a context-specific adoption of the design pattern to the software system, we argue that their comprehension plays a crucial role in creating and maintaining such correct implementations over the system's lifespan. Even with migration and integration of legacy components into an adaptive System, where other paradigms are used, for example, must be compatible on a conceptual level. Given a set of implementation samples, this paper intends to separate essential syntactic information from varying aspects. We introduce an approach that abstracts given object-oriented implementations by semantically resolving and splitting an Abstract Syntax Tree into small paths. The contribution this paper provides is composed of two parts. First, we introduce an approach to extract negligible details of given concept examples to distill the essence of concepts, and the second part presents a formal foundation to describe and interact with concepts. Based on this foundation, we derive several underlying problem statements.**

***Index Terms*—Software Architecture; Architectural Concepts; Design Pattern; Concept Extraction; Source Code Comprehension.**

I. Introduction

This is an extended journal paper extending the work presented in [1]. Design patterns have been established for reusing proven solutions to a class of problems. Nevertheless, especially for a dynamic adaptive system, the correct implementation of adaptation mechanisms is essential for the quality of the overall system. Patterns are described informally or semi-formally as context-independent solution concepts. As a consequence, in order to apply a design pattern, it is necessary to embed it into the actual implementation context; to do so, a common understanding of the concept provided by the pattern had to be established [2] [3].

To relate implementation and architecture, the Unified Modeling Language (UML), for example, offers the mechanism of collaborations within the context of a composition structure diagram and the context-specific embedding in a given domain. Here, the description is separated from the actual application in modeling. Collaborations describe the composition of roles, which must be linked to specific parts of the application [4] [5].

Faulty implementations of patterns may produce functionally correct solutions but may lack the (mainly) non-functional properties provided by the pattern, such as specific modularity goals or specifications from the software architecture [6]. Inaccurate implementations can emerge not only in the initial implementation of the pattern but also from side effects introduced with changes, even elsewhere in the codebase [7] [8]. In particular, in a scenario where system parts and components are implemented and maintained heterogeneously and by different companies and development teams, as is unavoidable in an adaptive Software Ecosystem, for example [9].

If a legacy system or component is to be migrated and integrated, for example, to satisfy a specific adaptation mechanism, it is necessary to check the current implementation's compatibility. For this, it is helpful to find design patterns in existing code to comprehend the whole system better. Especially if it is written by other developers or not further documented. With a focus on code comprehension, it is necessary to extract more complex architectural patterns from simple code patterns iteratively. As a starting point, this paper contributes to recognizing design patterns by generating a data-driven interpretable representation of the design pattern from a set of implementation examples and counterexamples. No formal specification of the design pattern beforehand is needed. This paper addresses the following **Research Questions (RQs)**: RQ1: *Is it possible to abstract different concrete implementations of the same architectural design pattern so that the abstractions show a similarity?* RQ2: *Is it possible to formulate what the shared concept consists of across multiple samples?* RQ3: *Is it possible to classify unseen samples using the introduced formulation mechanism?*

The contribution of the extension is the provided formal context on top of the introduced approach. On the one hand, this is necessary to work out the underlying problems and, on the other hand, to provide a formal foundation for further work. First, we introduce the terms and understanding of a **Concept** (Definition 1) and **Context** (Definition 2) in general. Then, based on the extensional description of sets, we define

an **Architecture Concept** (Definition 5) as a named Set of semantic equivalent examples. We introduce the **Abstract Syntax Graph (ASG)** (Definition 6) as an extension of the Abstract Syntax Tree (AST). Further on, we derive the term of an **Atomic Concept** (Definition 7) and a **Minimal Example** (Definition 10). Concerning the compositional characteristic of a concept, we propose a **Role** pattern (Definition 9). We close by introducing a **concept for instanziation** (Definition 11).

Furthermore, we have derived challenges from the proposed theory. We have outlined possible solutions to address them by introducing the so-called **Concept-Graph** (Definition 14) and deal with similarity instead of equality by defining the **Fuzzy-Hypergraph** (Definition 16) as an extension of the underlying graph representation.

Section II gives foundations on programming languages and the construction of the ASTs. Section III introduces the source code abstraction approach alongside two different levels of abstraction. Section IV is the evaluation of the stated RQs with a discussion of the results and limitations. Section V gives a formal approach for describing concepts. Section VI presents an overview of related work. Section VII opens challenges of extracting architectural concepts from given implementations. Finally, the conclusion and an outline of future work are given in Section VIII.

II. Foundation

This paper investigates the compositionality of abstract concepts. The inputs for the presented approach are syntactically correct but not executable source code artifacts. The focus is, therefore, on the static structure of a program. This structure is defined by the syntactic and semantic rules of a programming language. Each programming language consists of a set of programming concepts and specified paradigms, applying to modern programming languages that do not strictly follow one paradigm [10].

These concepts, defined by the programming language, are called **atomic concepts** (see Definition 7) in the following and manifest themselves in the source code by the language's **keywords**. Programming languages are formal languages because they consist of words over a given and finite alphabet [11]. Thus, the words are well-formed concerning a fixed and finite set of formal production rules [12]. Moreover, the lexical grammar of a programming language is usually context-free [13].

A grammar G consists of a four-tuple.

$$G(N, \Sigma, R, S) \tag{1}$$

with N : finite set of nonterminal symbols,
disjoint with the strings produced from G.
Σ : finite set of terminal symbols, disjoint from N.
R : finite set of production rules: $N \rightarrow (\Sigma \cup N)^*$
where * is the kleene star operator.
S : distinguished start symbol, $S \in N$.

We focus on object-oriented programming languages. Consequently, the type-system plays an important role and can be understood as an assurance to operations and documentation that can not be outdated. Types predefined by the programming language are so-called **atomic types**. Out of these atomic types, abstract types are constructed. The step of abstraction, which is also the foundation of the principle of information hiding, of abstract types is the structure defined by fields and an interface specified by the operations.

Since the languages considered here are formal, an automaton can be specified, which can process the character stream of the source code artifact. This is also the first step in compiling a program. Figure 1 shows the steps relevant to this paper of analyzing a program by a compiler. First, a scanner transforms the input stream into a language-specific token stream during lexical analysis. The tokens are also significant parts of a program, as they contain the atomic concepts of the programming language. This step reduces complexity, aggregates character, and identifies keywords. Then, a tree is generated from the token stream during syntactic analysis. A **tree** is a recursive data structure and a particular type of graph structure (a formal definition can be found in Section III-D) with a dedicated root node containing no cycles. Finally, each recognized token is converted to a node in the tree. Then, a semantic analysis is performed since not all rules, especially context-dependent ones, can be checked during derivation. This step also resolves the types, and names and annotates the tree's nodes to reflect this. Therefore, a symbol table is used to map each symbol with associated information like type and scope.

Through the instantiation of types, another kind of context-dependencies arises, which leads to the fact that the semantic meaning of a word derived by the grammar is no longer unique.

The challenge in extracting higher-level concepts up to architectural concepts is that these concepts are not included as concepts in the programming language. Instead, these can be understood as the composition of atomic concepts within a respective context. For program comprehension, it is essential to get a precise understanding of the concepts used in the implementation. Therefore with the increasing complexity and evolution of the program describing the essence of a concept in a comprehensible way to humans is a critical task.

It follows directly from the chosen class of language type that the set of generated concepts is countably infinite. Also, the set of reference implementations is infinite, with the difficulty that the same concept can be implemented in different ways. Thus, similarity could not be detected with a simple comparison of source code snippets.

III. Source Code representation

The main objective is a way to represent object-oriented source code samples on an abstract level compared to the raw source code files to enable interpretability on common parts and differences. Reducing information such as the naming of elements (e.g., methods, variables) or the order in which

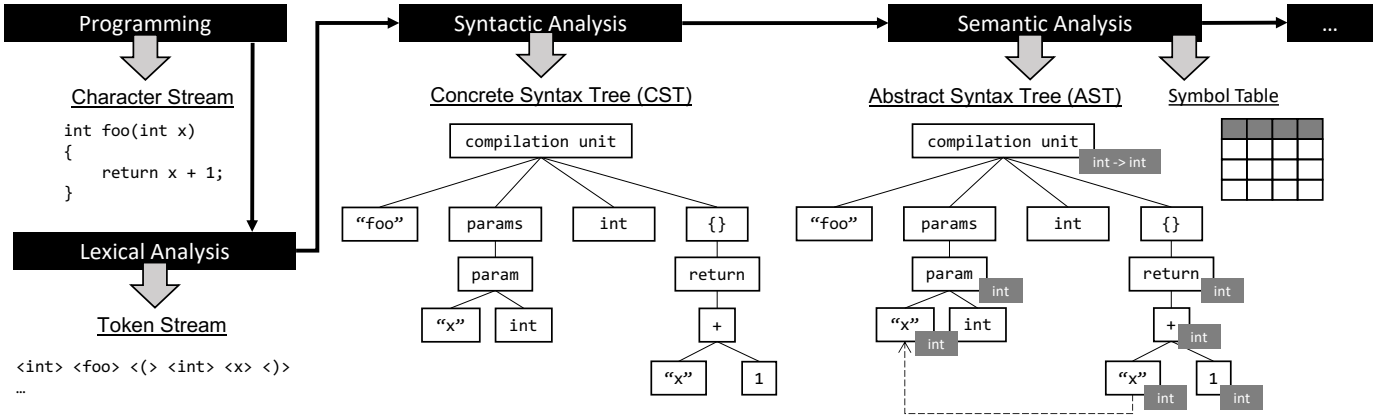


Fig. 1: First steps of a compilation process [13]

parts of the snippet (methods, variables) are declared or logic is handled (e.g., cases in a switch statement) help in this approach as it distracts from syntactical similarities.

We introduce two different levels of abstraction that both allow the expression of smaller parts reoccurring across different valid code snippets following the language's grammar rules. The **abstraction level** $High$ (Section III-B) is more abstract than level Low (Section III-C). The more concrete level of abstraction has superior expressiveness as it adds constraints across multiple reoccurring parts and allows for the distinction of elements (e.g., methods, variables).

We will elaborate on our general approach (Section III-A), being identical for both levels of abstraction first, then elaborating on $High$ (Section III-B), and adding in how we use the concept of uniquely identifying parts in Low. In Section III-C we explain how such constraints are added. In Section III-D we address how abstractions of different samples can be compared. Section III-E introduces the shared concept and how to construct it based on given code samples.

A. Source code abstraction approach

The approach, as illustrated in Figure 2, takes source code of arbitrary size as an input to generate an abstract representation in the form of a set of *Strings* that represent its syntax with additional information from the semantic analysis and aggregation. The Strings are sequences of tokens retrieved while processing the input that does not need to be exact sequences of the *Lexical Analysis*, as shown in Figure 1. A detailed walk-through example can be found in Sections III-B and III-C, Figure 1 contains only an illustrative one.

We analyze the code snippets *AST* to get a syntactic representation of the sample. The AST tokens get resolved during the aggregation phase constructing an *Aggregated Graph*. By combining the *ASTs* paths and the *Aggregated Graph*, we create the flattened *Abstraction*.

Subsequently, we formalize the required representations (AST, Graph, and the Abstraction) and concepts (path, aggregation function). Based on these definitions, we introduce the idea of a shared concept.

We define the **graph** $g \in$ GRAPH by the following signature:

$$g(V, E) := \{V = \{v_1, v_2, \ldots, v_n\}, E \subseteq V \times V\} \quad (2)$$

with V : finite indexed set of nodes.
E : finite indexed and ordered set of directed edges $\{v_i, v_j\}$

and a **tree** $t \in$ GRAPH being a special cycle-free graph with a root node v_{root} and a set of leaf nodes V_{leaf}

$$t(V, E, v_{root}, V_{\text{leaf}}) := \{g(V, E), v_{root}, V_{\text{leaf}}\} \quad (3)$$

$$\begin{aligned} \text{with } & V_{\text{leaf}} \subset V \wedge v_{\text{root}} \in V \\ & \forall v \in V \nexists v \mid \{v_{\text{root}}, v\} \in E \\ & \forall v_{\text{leaf}} \in V_{\text{leaf}} \nexists v \mid \{v, v_{leaf}\} \in E \end{aligned}$$

A path p in a tree t is a sequence of nodes V connected by edges E. The first node needs to be a leaf node and the final node needs to be the root node v_{root} of t.

$$p(V, E) := \{V, E\} \quad (4)$$

$$\begin{aligned} \text{with } & V := \{v_i \mid 1 \leq i \leq n\} \\ & v_1 \in t(V_{leaf}) \wedge v_n = t(v_{root}) \\ & E := \{\{v_{j-1}, v_j\} \mid 2 \leq j \leq n\} \end{aligned}$$

In the *Aggregation* step, the nodes of the AST get mapped to nodes of a resulting *Aggregated Graph*, by an aggregation function $f_{\text{aggregate}}(t) := V_t \rightarrow V_g$.

To construct the abstract representation a a concrete aggregation function combines the information of all paths P of the tree and the graph g itself. P is the set of paths containing each path from every leaf node of V_{leaf} to the root node v_{root}. It is defined by the following signature:

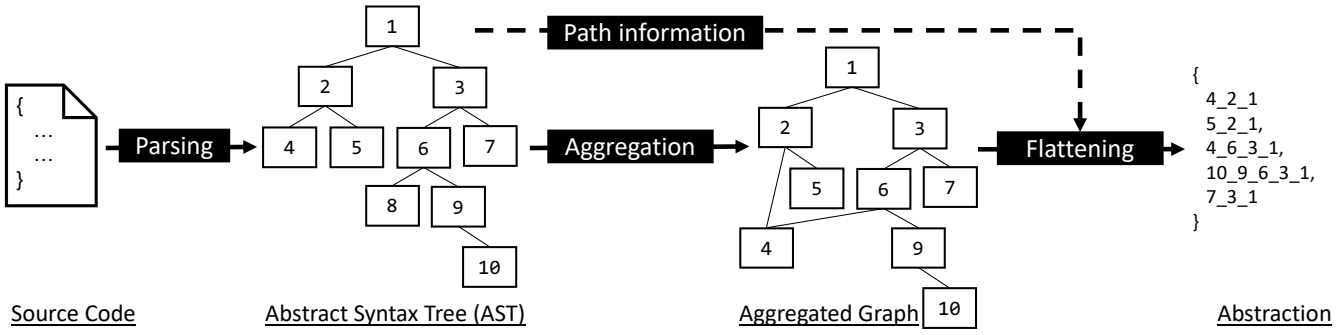


Fig. 2: Overall approach of the source code abstraction

```
1 public class FooBar {
2         public void foo() {…}
3         public void bar() {…}
4 }
```

Fig. 3: Java implementation of a class with two methods - *program 1*

$$P := \{p \mid p(v_1) \in t(V_{\text{leaf}}) \wedge p(v_n) = t(v_{\text{root}}) \wedge \forall v_{\text{leaf}} \in t(V_{\text{leaf}}) \exists! p \mid v_{\text{leaf}} \in p(V)\} \quad (5)$$

An **abstraction** is defined by the function f_{abstract} :

$$f_{\text{abstract}}(t, f_{\text{aggregate}}(t)) := (V_t, E_t) \times (V_g, E_g) \rightarrow P \quad (6)$$

To obtain the flattened abstraction, we combine the path information from the tree and the node information from the aggregated graph. The structure of the flattened Strings in the abstraction comes from the Paths P in the AST. The information of the relevant nodes results from applying the $f_{\text{aggregate}}$ function to the nodes of the paths $p \in P$. The final abstraction is a set of all distinct flattened Strings. In the example Figure 2, the aggregation merges the nodes *4* and *8* (from the AST). Those nodes represent the same semantic unit (e.g., the same literal) In this case p is *"8_6_3_1"*, after applying $f_{\text{aggregate}}$ the flattened String is *"4_6_3_1"*.

B. Abstraction level $High$

The nodes (tokens) in an AST have additional traits. We utilize the type of the node, which indicates what part of the language the node reflects (e.g., the declaration of a class or the call of a method). In addition, we use the information of more basic nodes (e.g., keywords, primitive operators) to represent individual nodes per manifestation (e.g., *TRUE* and *FALSE* for *Boolean* values) and one node per *Modifier* (e.g., *PRIVATE, PUBLIC, and STATIC*). On $High$, the aggregation step summarizes all nodes of the same type (e.g., all nodes that declare methods) into a single node.

Figure 3 shows a short code snippet that we will use for both abstraction levels to illustrate the approach and the resulting representations. The sample consists of a *public class FooBar* containing two methods (*foo* and *bar*). The content of the methods is left out, as it would be hard to display the resulting ASTs and graphs. As illustrated in Figure 2, we start with traversing the AST. The resulting tree is shown in Figure 4. In the tree, we can see the individual statements reflected by nodes and corresponding edges. Each node contains the information of the type of the node (e.g., *ClassDeclaration* for the root element) and, if available additional information such as the reflecting values associated with the nodes (e.g., *SimpleNames* reflecting the name of the class *FooBar* and the names of the methods *foo* and *bar*) or the proper modifier (in this case *PUBLIC* in all instances).

The higher-level **aggregation rules** of nodes are: (i) resolve keywords from the language. This includes *Primitive Operators, Primitive Types, Modifiers, TRUE, FALSE*, and (ii) reduce other nodes to the assigned types.

Figure 6a shows the resulting graph by applying the aggregation rules. Our abstraction aims to (i) consist of multiple small parts (ii) likely to be contained in multiple samples. From the tree (Figure 4), the graph (Figure 6a), and the aggregation rules , it is possible to construct the paths in Figure 5. Here underscore separates the nodes in a flattened path.

Carried information $High$: The paths extracted carry certain information enabling reasoning about the original program. For example, the second path states that there is a *PUBLIC ClassDeclaration* (line 1 of the code sample in Figure 3). The third path states a *PUBLIC MethodDeclaration* in a *ClassDeclaration*. From the information contained in the abstraction, we cannot tell which methods *foo* or *bar* this particular path represents.

On $High$, we cannot conclude across multiple paths. For example, it is impossible to state that the *MethodDeclaration* from paths 3 and 4 are part of the same *Method*. On the one hand, this shows that the abstraction level is capable of reflecting the general structures of the original code while being able to ignore the order of appearance in the original implementation. On the other hand, the abstraction lacks the distinction of different elements and the ability to connect multiple paths related to each other.

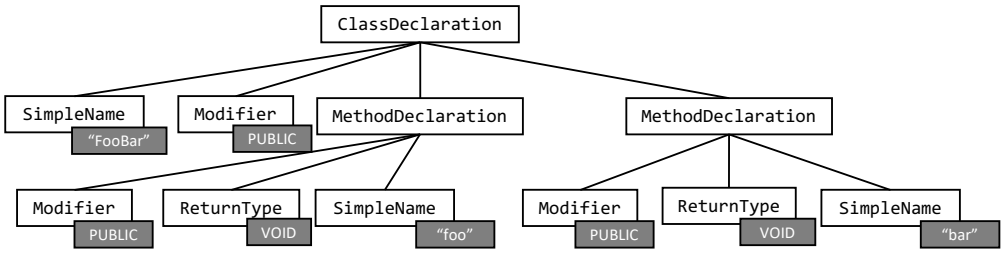


Fig. 4: AST of program 1

```
1 SimpleName_ClassDeclaration
2 PUBLIC_ClassDeclaration
3 PUBLIC_MethodDeclaration_ClassDeclaration
4 VOID_MethodDeclaration_ClassDeclaration
5 SimpleName_MethodDeclaration_ClassDeclaration
```

Fig. 5: Abstraction $High$ of program 1

C. Abstraction level Low

The stated drawbacks of $High$ get addressed at Low, containing more information from the original sample. The overall approach (Figure 2) still holds, with different steps in the aggregation phase. Semantic analysis of the AST is utilized to resolve elements. We introduce indices to those resolved elements, allowing the distinction of multiple nodes (of the same type and even across multiple types). The **aggregation rules** are as follows: (i) exactly as the first rule on $High$; (ii) identification of *Classes* and *Methods* by their signature; and (iii) resolution *(Simple)Names* with an index per unique name.

According to the stated rules, aggregation of the AST leads to the graph illustrated in Figure 6b. The indices allow the identification of elements. For example, we can still refer to the methods using index 1 and 2. The index is attached in the flat representation of the paths, separated by a hash symbol. The resulting paths of the code sample on Low are given in Figure 7. All the information of $High$ is still contained in this representation, as it is possible to remove all the indices and remove the duplicated paths resulting in Figure 5.

Carried information Low: The indices allow (i) to conclude across multiple paths, (ii) to distinguish multiple elements of the same type (e.g., the two *Methods*), and (iii) to express constraints that join different types seen in the aggregation process to superior entities (e.g., using one index for a specific *MethodDeclaration* and *MethodCallExpression*).

In Figure 7, all the paths are in the context of the same *ClassDeclaration(#1)*. We can draw conclusions about *MethodDeclaration(#1)* from paths 3 and 4 and state that it is *PUBLIC* and has the return type (*VOID*). The same holds for paths (6 and 7 respectively for the second *MethodDeclaration*). To distinguish elements across multiple paths the indices can be used similarly. We can tell that paths 5 and 6 are not belonging to the same *MethodDeclaration*.

D. Abstraction alignment

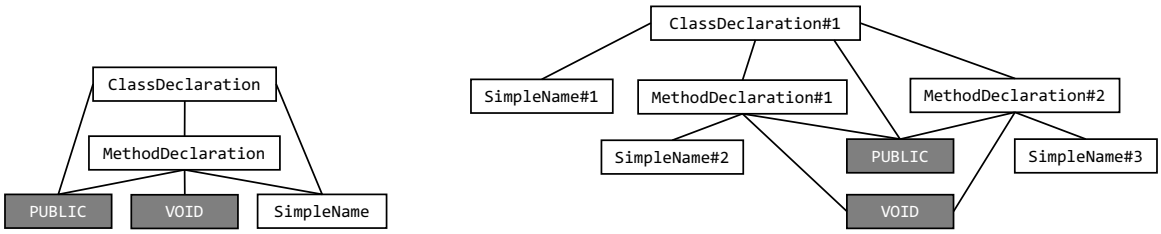
In the sections above, we introduced abstraction levels $High$ and Low for one single code snippet, both providing a set of paths representing the snippet. We showed how to reason across multiple paths of one abstraction. The next step in making use of the representation is to reason across multiple abstractions of different snippets x and y, by considering the sets of paths P_x and P_y, respectively, that they generate. We propose a *Jaccard Similarity* (Formula 7) based measurement, leading to a high similarity if a lot of paths are in both sets P_x and P_y, and little paths only in either set P_x or P_y.

$$jaccardSim(P_x, P_y) := \frac{\mid P_x \cap P_y \mid}{\mid P_x \cup P_y \mid} \tag{7}$$

On $High$ it is easy to be calculated without further steps needed, as no instance (e.g., multiple methods) are distinguished. On Low, the calculated similarity will depend on the indices assigned to the individual parts in the aggregation step, as the following example in Table I illustrates. The table is two parts, with the upper part containing different paths (left-hand side) and three abstractions (P_a, P_{b1}, and P_{b2}). An x in the respective cell means that the path is part of the abstraction. The lower part of the table contains the pairwise Jaccard similarity. The similarity calculated differs between *jaccardSim*(P_a, P_{b1}) and *jaccardSim*(P_a, P_{b2}) regardless of both P_{b1} and P_{b2} being equally valid representations of a *Class* having one *PRIVATE* and one *PUBLIC Method*.

In the presented approach (Figure 2) the indices get assigned in order of node processing. If a node (e.g., a *MethodDeclaration*) has been seen before, the assigned index is reused, otherwise, the next available index (per node type) gets assigned. This could lead to P_{b1} or P_{b2} for the same code sample, that are equally valid abstractions.

The idea to counteract this is by aligning the samples to improve the similarity measured without alternating the information contained in the abstractions. We achieve this by looking for (sub)graph isomorphism and corresponding permutations. In this example, a similarity-maximizing permutation of P_{b2} regarding P_a would be to swap the indices of the two *MethodDeclarations*. An important remark is that such a swap of indices needs to conform to the **permutation rules** (i) the swap of indices needs to be done for all occurrences to not invalidate a constraint and (ii) entities need to be respected, so the index of such related types need to be aligned uniformly.



(a) Abstraction level $High$

(b) Abstraction level Low

Fig. 6: Resulting graphs by aggregating nodes and edges of the example AST

TABLE I: SAMPLE ABSTRACTIONS AND CORRESPONDING PAIR-WISE JACCARD SIMILARITIES

paths on low abstraction level	P_a	P_{b1}	P_{b2}
PUBLIC_ClassDeclaration#1	x	x	x
PUBLIC_MethodDeclaration#1_ClassDeclaration#1	x	x	
VOID_MethodDeclaration#1_ClassDeclaration#1	x	x	x
PRIVATE_MethodDeclaration#1_ClassDeclaration#1			x
PUBLIC_MethodDeclaration#2_ClassDeclaration#1			x
VOID_MethodDeclaration#2_ClassDeclaration#1		x	x
PRIVATE_MethodDeclaration#2_ClassDeclaration#1		x	
jaccardSim with P_a	1	0.6	0.33
jaccardSim with P_{b1}	0.6	1	0.429
jaccardSim with P_{b2}	0.33	0.429	1

```
1 SimpleName#1_ClassDeclaration#1
2 PUBLIC_ClassDeclaration#1
3 PUBLIC_MethodDeclaration#1_ClassDeclaration#1
4 VOID_MethodDeclaration#1_ClassDeclaration#1
5 SimpleName#2_MethodDeclaration#1_ClassDeclaration#1
6 PUBLIC_MethodDeclaration#2_ClassDeclaration#1
7 VOID_MethodDeclaration#2_ClassDeclaration#1
8 SimpleName#3_MethodDeclaration#2_ClassDeclaration#1
```

Fig. 7: Abstraction Low of program 1

The **isomorphism** between two graphs is a *bijection* (one-to-one correspondence) between the nodes of the given graphs. As the graphs in our case are not guaranteed to be of the same size, we need to look into subgraph isomorphisms of the size of the smaller graph. A **subgraph** m of a graph g is denoted by:

$$m \subset g \iff V_m \subset V_s \wedge E_m \subset E_s \tag{8}$$

Finding such a *bijection* (candidate) of a subgraph consists of two steps, (i) fixing a suitable subgraph and the (ii) one-to-one correspondence. The verification of such a candidate can be done with Formula 9. The graphs q and m are converted to adjacency matrices (see Formula 10), and the *bijection* is formulated as a **permutation matrix** Q. Q is constructed with the nodes of one graph as rows, and nodes of the other graph as columns, the cells representing a correspondence are filled with 1, all others with 0. An adjacency matrix D_m contains a row and column for each node of the graph m, the respective cell is filled with 1 if there is an edge between those nodes, with 0 otherwise.

Let q be a graph isomorphic to m, for some *permutation matrix* Q:

$$q \cong m \iff \exists Q, D_m = Q \times D_q \times Q^T \tag{9}$$

Let D_m be the **adjacency matrix** of m, with:

$$D_m ij := \begin{cases} 1 & \text{if } \{i,j\} \in E_m \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

After an isomorphism has been found, the indices can be aligned according to the permutation, allowing for the final check to see if the resulting paths match. This is needed as g (and D_m) do not contain the information of the original paths, so the graph will accept possible paths not contained in the abstraction.

E. Shared concept

We define a shared concept c_{shared} as the set of similarities and differences between a set of code snippets. The abstractions of code snippets, which contain the concepts c_{shared} are elements of the set A_{in} and code snippets, which are not an implementation of the concept c_{shared}, represent an element of the set A_{ex}.

Out of these two sets of abstractions of examples and counterexamples, the representation of the shared concept is derived as follows:

$$c(A_{in}, A_{ex}) := \{P_{in}, P_{ex}\} \tag{11}$$

$$\begin{aligned} \text{with } & P_{in} \cap P_{ex} = \emptyset \\ & \forall p_{in} \in P_{in} \wedge \forall a_{in} \in A_{in} \mid p_{in} \in a_{in} \\ & \forall p_{ex} \in P_{ex} \exists a_{ex} \in A_{ex} \mid p_{ex} \in a_{ex} \\ & \forall p_{ex} \in P_{ex} \nexists a_{in} \in A_{in} \mid p_{ex} \in a_{in} \end{aligned}$$

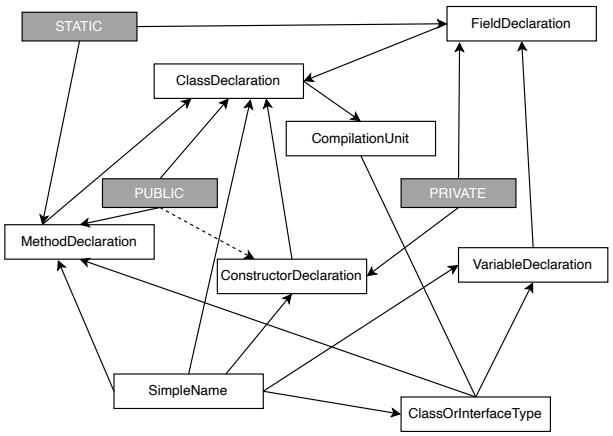


Fig. 8: Graph reconstructed from the P_{in} paths

Related to the above definition, a shared concept is described by two sets of paths P_{in} and P_{ex}. Each path $p_{in} \in P_{in}$ is included in every single abstraction of A_{in}. P_{ex} consists of paths p_{ex} retrieved by the set of abstractions A_{ex}. For a path to be included in P_{ex} it needs to be in at least one abstraction of A_{ex} and must not be in any abstraction of A_{in}. The idea of those exclusion paths is to handle paths seen in the programming language that have never been seen in a positive example that is expected to include the shared concept. By including samples from different repositories and business domains into the sets A_{in} and A_{ex} we hypothesize that the shared concept is containing business-domain-independent overlap.

After defining the shared concept in terms of P_{in} and P_{ex}, we can now reconstruct a graph based on the set of examples. We could obtain different graphs containing details of (i) all examples, (ii) only the examples A_{in}, or (iii) only examples A_{ex}, or even more fine-grained graphs. To construct the graph, we reverse the flattening step shown in Figure 2 in the overall approach. Note, that we are now working on the common paths of multiple examples and are constructing a graph that is not based on a single source code sample anymore. Figure 8 shows the newly constructed graph primarily based on P_{in} (Figure 9). Containing all the nodes and (solid) edges. In addition, the graph contains (dotted) edges coming from P_{ex} (Figure 10), which are between nodes that are present from P_{in} and not already present through P_{in}.

We hypothesize that the resulting graph contains the common entities and their syntactic relationships and other known syntactic relationships of the programming language that are not part of any example of the concept.

IV. Evaluation

The evaluation starts with describing the data set, which was collected and annotated by the authors. The second part introduces the singleton design pattern, as this is the case study through the evaluation of the paper. The rest of the section addresses the stated RQs. We start by finding similarities on the abstraction levels (RQ1), calculating pair-wise Jaccard similarities on the abstraction levels, and analyze how the similarity compares on pairs that are both singletons, one of the samples being a singleton and non of the samples being a singleton. We formulate the shared concept as RQ2, by including all paths P_{in} we have seen in all samples (of the singleton), in addition, we formulated an exclusion set of paths P_{ex}, by specifically excluding paths that we have only seen in non-singleton samples.

Classifying new samples on the abstraction levels using the formulated shared concept (RQ3) is done as the last part of the evaluation.

TABLE II: ANALYSIS OF THE AMOUNTS OF PATHS IN THE ABSTRACTIONS

	min # of paths		max # of paths		avg. # of paths	
	low	high	low	high	low	high
singleton	17	17	2379	646	247.26	88.61
non singleton	6	4	2856	983	421.16	157.37
all samples	6	4	2856	983	334.21	122.99

A. Results

1) Preprocessing of the data set: The data set (*java-singleton*) collected and used to evaluate the abstraction approach consists of 230 java code samples labeled as part of this paper, containing the singleton design pattern and 230 additional samples that do not implement the singleton design pattern. The classes originate from different projects. The labels were applied by two authors, only containing samples confirmed by both authors. We chose the singleton pattern as a concept to evaluate as it combines a few criteria we consider beneficial as a showcase in this paper. The purpose of the pattern is widely understood and used in practice. The implementation is all in one place (the singleton class), leaving aside large search spaces [14]. Making it reasonable to identify samples in existing code but leaving room for the implementation to vary. It introduces manageable complexity to the task at hand while enabling us to collect a data set to evaluate the presented work, although the presented abstraction approach is not limited to the scope of a single class, file, or pattern. We abstracted all the samples on both levels of abstraction. Table II gives insights into the resulting abstractions. The table contains the minimum, maximum, and average amount of paths for all abstractions of a given set of abstractions. The sets show that the range of how many paths are in the samples varies a lot for each given set inspected. The average is also significantly higher than the minimum amount of paths. This indicates an overlap exists, and the samples have something to do with each other.

2) Results RQ1: As described in Section III-D we are going to measure similarity using the *Jaccard Similarity* (Formula 7). Table III summarizes details on the calculated similarities. Each row represents ten percent incremental thresholds, with the corresponding amount of sample pairs that are at least as similar as the threshold requires. The reported numbers are broken down into how many pairs are (i) both singletons, (ii) one of them is a singleton and, (iii) none of them is a singleton.

TABLE III: NUMBER OF SIMILAR PAIRS ABOVE 10 PERCENT INCREMENTAL THRESHOLDS

	Low			$High$		
threshold	both singleton	one singleton	none singleton	both singleton	one singleton	none singleton
0.0	26335	52900	26335	26335	52900	26335
0.1	4843	389	31	22628	21859	7950
0.2	1444	4	4	10395	1630	600
0.3	372	0	1	3737	118	73
0.4	135	0	1	1589	9	28
0.5	73	0	0	669	0	16
0.6	43	0	0	289	0	8
0.7	32	0	0	153	0	1
0.8	28	0	0	95	0	1
0.9	27	0	0	63	0	0
1.0	25	0	0	30	0	0

This is done for both abstraction levels. The comparison of the samples with themselves is excluded from the table.

The data shown in the table support the assumption that the abstractions embody similarities related to the singleton design pattern. From the columns *both singleton* on both abstraction levels, we take that the stated RQ1 holds and that it is possible to abstract different concrete implementations of the same design pattern to show a similarity. As the similarity observed is significantly higher compared to the other columns in the table.

3) Results RQ2: We built a shared concept as introduced in our Definition 11. This part of the evaluation is limited to $High$ as no complete alignment of all samples has been calculated, leading to inaccurate results on Low. More on this is addressed in the limitations and future work section of the paper.

We follow common practice in Natural Language Processing (NLP) (compare stop word removal [15]) and trim the data so that we do not rely on too (un)common paths. We only keep paths in at least 5 percent and at most 95 percent of the samples of the dataset.

Table IV distinguishes the (non-)trimmed abstractions. It displays the number of paths belonging to specific subsets of the data set. For the non-trimmed row, many paths are exclusive to (non-)singletons ($4644 + 12813$) compared to the 1996 paths shared. As the collected data set is small, contributing to infrequently observed paths, we focus on the trimmed column of the table. There are no paths left that are exclusive to the singleton samples. This allows us to ascertain, that there are no language constructs exclusively used to implement the singletons. In addition, eight paths are exclusive to non-singleton samples, which indicates that they are part of the programming language but not used to implement the singleton design pattern. No paths are seen across all non-singleton samples. The majority of paths are seen across both singletons and non-singletons. The shared concept retrieved from the data set *java-singleton* consists of twelve paths in P_{in} and eight paths in P_{ex}.

4) Results RQ3: To evaluate if it is possible to use the shared concept for classification of unseen code, we use a dataset [16] providing annotations of used design patterns. The dataset contains annotations for the following nine java projects: *QuickUML 2001, Lexi, JRefactory, Netbeans, JUnit, MapperXML, Nutch, PMD,* and *JHotDraw*.

```
1 STATIC_MethodDeclaration_ClassDeclaration_CompilationUnit
2 PUBLIC_MethodDeclaration_ClassDeclaration_CompilationUnit
3 SimpleName_VariableDeclarator_FieldDeclaration
    _ClassDeclaration_CompilationUnit
4 SimpleName_ClassOrInterfaceType_VariableDeclarator_FieldDeclaration
    _ClassDeclaration_CompilationUnit
5 SimpleName_MethodDeclaration_ClassDeclaration_CompilationUnit
6 SimpleName_ConstructorDeclaration_ClassDeclaration_CompilationUnit
7 PRIVATE_ConstructorDeclaration_ClassDeclaration_CompilationUnit
8 SimpleName_ClassOrInterfaceType_MethodDeclaration
    _ClassDeclaration_CompilationUnit
9 STATIC_FieldDeclaration_ClassDeclaration_CompilationUnit
10 SimpleName_ClassDeclaration_CompilationUnit
11 PRIVATE_FieldDeclaration_ClassDeclaration_CompilationUnit
12 PUBLIC_ClassDeclaration_CompilationUnit
```

Fig. 9: P_{in} paths on abstraction $High$ as shown in Table IV

```
1 SimpleName_NameExpr_MethodCallExpr_ObjectCreationExpr_ReturnStmt
    _BlockStmt_MethodDeclaration_ClassDeclaration_CompilationUnit
2 SimpleName_ClassOrInterfaceType_ClassOrInterfaceType
    _ClassDeclaration_ClassDeclaration_CompilationUnit
3 SimpleName_ClassOrInterfaceType_ObjectCreationExpr_ReturnStmt
    _BlockStmt_MethodDeclaration_ClassDeclaration
    _ClassDeclaration_CompilationUnit
4 SimpleName_MethodCallExpr_MethodCallExpr_MethodCallExpr
    _MethodCallExpr_ExpressionStmt_BlockStmt_MethodDeclaration
    _ClassDeclaration_CompilationUnit
5 SimpleName_NameExpr_MethodCallExpr_MethodCallExpr
    _MethodCallExpr_MethodCallExpr_ExpressionStmt_BlockStmt
    _MethodDeclaration_ClassDeclaration_CompilationUnit
6 PUBLIC_ConstructorDeclaration_ClassDeclaration_CompilationUnit
7 PUBLIC_ConstructorDeclaration_ClassDeclaration
    _ClassDeclaration_CompilationUnit
8 SimpleName_MethodCallExpr_ObjectCreationExpr_ReturnStmt
    _BlockStmt_MethodDeclaration_ClassDeclaration_CompilationUnit
```

Fig. 10: P_{ex} paths on abstraction $High$ (trimmed) as shown in Table IV

The authors of this paper validated the annotations. From the 13 annotations, we rejected seven, finding six additional singleton implementations that were not annotated as such before. Resulting in a total of 12 instances.

We conducted three experiments (Table. V)(i) *High incl.* only looking to include all the P_{in} paths, (ii) $High$ refers to in addition looking that none of the exclusion paths P_{ex} are present, and (iii) Low we used the inclusion paths P_{in} and

TABLE IV: SUB SETS OF THE DATA SET AND THE AMOUNT OF THEIR EXCLUSIVE PATHS

	# paths only in		# paths in all		# paths seen
	singletons	non-singletons (P_{ex})	singletons (P_{in})	non-singletons	in both sets
trimmed	0	8	12	0	279
not trimmed	4644	12813	12	0	1996

associated indices that conform to the singleton pattern. Here we then aligned the indices of the samples (using subgraph isomorphism).

As a given sample can be classified containing a singleton (*Positive*) or not (Negative) and the ground truth label can tell if it is a singleton or not, we end up with the resulting combinations *True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN)*. In our context, the classes mean: *TP*: prediction and ground truth agree on singleton; *TN*: prediction and ground truth agree on non-singleton; *FP*: prediction says singleton but it is not a singleton; and *FN*: predict says non-singleton but it is a singleton. To evaluate the performance of our classification of unseen samples we stick to the metrics of a confusion matrix used for the evaluation of Machine Learning (ML) models. Table V shows the results of the conducted experiments. Calculations of *Precision* also known as *Positive Predictive Value (PPV)*, *Recall* also known as *True Positive Rate (TPR), Accuracy (ACC)*, and *F1* are also calculated. A general remark is that the files were not changed or preprocessed. In the case of data set *java-singleton*, we isolated one class per code sample, contrarily those files used for the prediction are still untouched and possibly contain multiple classes.

B. Discussion

We have seen that abstractions produced by samples of various origins (different projects) carrying the same design pattern still carry a certain degree of similarity on the different levels of abstraction introduced in this paper. In terms of formulating the shared concepts, we were able to formulate a set of paths included in all samples and exclude a set of paths that we have only seen in other implementations that do not contain the same design pattern in the first place. The inclusion set P_{in} contains twelve paths, and the minimum number of paths seen in the set of singletons (see Table II) is only 17. This allows drawing the conclusion that at least one sample contains almost the bare minimum needed to implement a singleton in Java.

The exclusion set P_{ex} serves another important purpose, as it helps to explicitly describe what should not be part of the concept. In the case of the conducted evaluation, we reduced the exclusion set by trimming all paths that were in less than five percent of the samples, which allowed us to reduce the set from 12813 to only eight paths. We argue that this is useful because of the rather small sample size. We have not found another approach that similarly describes a concept by explicitly stating what is not part of the desired concept. Paths contained in P_{ex} were contrary to the definition of a singleton, as they contain paths for *Public Constructors*, and paths for creating new objects in the return statement of a method (which would bypass the singleton object, if it would be the *getInstance* method).

Also, the approach of the formulation of such a shared concept is flexible and adapts to the considered samples, and the more the samples share, the more is included. As the paths are interpretable, the abstraction levels introduced in this work also allow a formulation of such shared concepts from scratch, or to use only one example as a template to start with.

TABLE V: RESULTS OF THE PREDICTION TASKS

	TP	TN	FP	FN	TPR	PPV	ACC	F1
$High$ *incl.*	12	1914	13	0	1.0	.48	.993	.649
$High$	8	1919	8	4	$.\overline{6}$	.5	.994	.571
Low	12	13	0	0	1.0	1.0	1.0	1.0

Both run on $High$ have a PPV around 0.5, while the TPR is higher, not making use of the exclusion paths P_{ex}. The ACC of both approaches is also nearly identical at 0.99. Caused by the data having a lot of *Negative* cases, in which both approaches are good at predicting. By comparing both runs, it is indicated that $High$ lowers the prediction of singleton (TP and FP) while introducing FN. The last part of the evaluation has been performed on Low. In this case, we introduced indices to the paths in P_{in}. We then aligned the indices of the samples, according to a valid permutation. The results have a PPV, TPR, and ACC of 1. This classification task was only performed on the 25 samples predicted as *TRUE* on the most permissive other approach (*High incl.*). For two main reasons, (i) the computation needed to find a subgraph isomorphism is NP-complete [17], and (ii) the previous check on $High$ for all P_{in} excludes all the other samples for not having all the needed paths. By knowing not all paths are present in the other samples (regardless of indices) it is not possible to find indices for those samples so that all paths are included afterward.

In terms of the classification performed, we have shown predictions with simple models, checking the exact inclusion and exclusion of specific paths on the $High$ and the same thing (after the computational intense subgraph isomorphism checking) on Low, with a perfect result as a reward. The prediction on $High$ is prone to overestimate the concept to be included, which is indicated by a precision around 0.5 for the not preprocessed unseen samples. Nevertheless, $High$ serves a valuable purpose in filtering the relevant samples to further look at Low.

C. Limitations

Although the approach introduced gives promising results in terms of the stated RQs, we have encountered some limitations on which we want to elaborate.

The design pattern chosen is rather simple in terms of the variety the implementation offers. Looking at more complex structures (e.g., using general parts and specific refined parts could implement those as interfaces or (abstract) classes), in terms of the shown abstraction levels this would lead to not being reflected in P_{in} as of the current approach on building the inclusion set.

Assigning index-values to the shared concept Low was the only time (except the labeling) we relied on understanding the concept (of the singleton). To address that, the indexing can be seen as the maximum common subgraph problem [18] (being NP-Hard [17]). We do not have an implementation of this in our prototype.

V. A formal Approch for Describing Concepts

In this section, we discuss the term concept in general, and then a definition is derived for an architectural concept, which forms the foundation for this paper. The ”correct” naming of concepts is a very critical aspect. Therefore, first, the difficulty of naming ”things” from the perspective of natural language is discussed. Finally, the relation between the challenges of natural language, set theory, and graph theory is established.

A. Introduction of Concepts

The word concept is part of everyday language usage. It is applied in different contexts and domains. The term concept is generally defined as *”An idea or a principle that is connected with something abstract”* [19]. The term concept comes from the Latin word *concipere*: to put together, to formulate, to comprehend. From these definitions, essential characteristics that make up a concept can be derived.

The essential characteristic properties of a concept are:

1) made by people for people.
2) does not have to be realizable.
3) describes something abstract with the aim of comprehending a fact.
4) is context-independent and can therefore be applied in different contexts.
5) can be described in different ways.
6) can arise from concepts by hierarchical composition.
7) does not have to be explicit.

A concept has in common with a software architecture that people set it up as a communication instrument to create a shared understanding. Abstraction is an essential characteristic of a concept, as it is for software architectures. A realization, an example existing in reality, does not have to be present. A concept description is incomplete and limited to the essential elements of the concept. Whereby a concept, in contrast to the architecture of a software system, can be considered context-independent and has, therefore, more of the character of a reusable pattern. Although concepts can be applied in different contexts/architectures, they must be refined and adapted to the context.

Another common feature between architecture and a concept is the variety of description techniques. Everything is present, from natural language texts to complex, even executable description techniques. In the Latin origin of the word, it is already clear that a concept is something composed. Accordingly, the components of a concept play an essential role in the description and creation of shared knowledge. If a concept is compositional, it follows that there are different levels of abstraction, which have no global but only a relative reference. Implicit in the word definition, expressed by the synonyms plan, sketch, and draft is the intention to accomplish a task or solve a problem. Detached from any domain and referring to the previously listed characteristics, the term concept is defined as follows:

*Definition 1 (**Concept**): A concept is a context-independent, abstract description of a schematic solution to a class of problems generated by people for people.*

A concept can be described in two ways. On the one hand, by the description of the partial concepts and their composition or on the other hand by giving an example in the form of a sketch or an exemplary structure for concepts, which have an implementation in the real world. Both approaches have advantages and disadvantages and a direct correspondence to the extensional and the intensional definition of mathematical sets [20]. Describing by example is generally easier to understand but does not guarantee that the concepts' intention in all its manifestations has also emerged. However, a concept specification may result in a lack of reference to the real world and make it challenging to apply the concept. To understand the concept, an awareness of how it ”works” must be created. This can be done either through a clever selection of examples or an appropriate specification. Generally, these properties only come into play during composition and application in a concrete system (cf. [21] [22]).

In mathematics, the underlying principle is called conceptualism. According to this, mathematical terms are not fixed; they develop. The handling of the definitions in practice, the interaction with other mathematical concepts, and the exchange between people influence this development. This evolution of terms and mathematical concepts can lead to an increasingly uniform usage and, as in the case of set theory, to axiom systems. (cf. [23])

The definition and thus also its representatives can therefore change over time. How does a concept arise? – According to the identified characteristics, it arises by an extensional or intensional set definition, exactly when elements are composed in such a way that they contribute in at least one context to a problem solution. A set is intensionally described by the specification of a property that can be assigned to that set. An intensional definition in the context of free and arbitrary choice of property, antinomies, can be induced; an example is Russel's antinomy [24]. Such antinomies are met with an appropriate axiomatization so that the underlying universe is well-defined. An axiomatization for sets is the Zermelo-Fraenkel set theory [20]. In this, predicates are represented by sets, namely subsets of the powerset, constructed via the scheme of the axiom of elimination. If properties construct sets and concepts can be defined over sets, then properties must

exist which represent concepts. Consequently, subsets of the power set exist with elements that fulfill a specific property and represent a concept.

It is easy to see that the essence of the concept must result from the semantic interpretation of its elements. Regarding the definition, a characteristic feature of a concept is schematic problem-solving. This is something that is not fundamentally true for properties. For a property, according to Platonic logic and as for the symbol $\in$, it is true that any object of a well-defined universe has a property or not. Additionally, the nature of a property does not change over time. From an epistemological point of view, this does not apply to concepts, as described earlier, for the principle of conceptualism. Because of this and the problem-oriented character, a concept can be understood as an embedding in an element to fulfill a particular property.

If elements are related to other elements, new properties can arise through emergence, attributed to the composed element. If a property becomes significant, then a property becomes a concept exactly if this property is additionally attributed to a problem-solving character. Both the significance and the solution of a problem are in the eye of the beholder and thus depend not only on the individual but also on the context.

According to the definition, a concept is a context-independent description. Thus, the application of a concept is its embedding in a concrete context. The word context comes from the Latin word contextus, close linkage, and relationship and can be generally defined as follows.

*Definition 2 (**Context**): A context represents a technical or situational setting that is meaningful for purposes and comprehension.*

B. The Difficulty of Naming Things

Another aspect that comes into effect when people interact with each other is verbalization, which becomes even more critical through context-sensitive embedding. This section explains why it is difficult to name things unambiguously so that communication partners have a shared, and equal understanding. It also discusses the differences and similarities between natural and formal languages, such as programming languages.

In linguistics, the term concept is used as the cognitive representation of an object or a cognitive category and is closely related to the meaning of a word [25]. This often makes it difficult to separate the concept from the context. It is not uncommon for words in a natural language to have different meanings. The meaning of a word arises from the context in which it is used. From this context, not only the meaning arises, but consequences are also connected, which are valid only in this context. In linguistics, the context is understood as the surrounding text of a linguistic unit, although it is not excluded that certain meanings remain open.

For most natural languages, words are formed by concatenation over an alphabet, given the construction rules of a concrete grammar. In a formal language, the language is entirely described by the grammar. However, natural languages are generally not formal languages. In this case, meaning is created only by combining letters to form words. Hence words are also referred to in linguistics as the atomic concepts of a language [26]–[28].

*Definition 3 (**Atomic Concept**): An atomic concept is a concept that is part of the language used to name realizations of the concept.*

It follows that atomic concepts can be clearly identified as part of language, as they can occur directly in the rules of grammar but are defined in dictionaries, especially in the case of natural languages. In linguistics and psychology, the meaning of complex expressions is attributed to the compositionality of concepts of language [28]. Thomas Ede Zimmermann describes the ordinary principle of compositionality as follows: *"The meaning of a complex expression functionally depends on the meanings of its immediate parts and the way in which they are combined."* from [29], p. 3 This definition goes back to the teachings of Aristotle and is known as the semantic compositionality principle (Frege principle). Named after the German mathematician Gottlob Frege (after [30] [31]).

The only fundamental assumption underlying this principle is that the atomic parts, which cannot be further decomposed, have a lexical meaning. Atomic concepts, like properties, have a stable meaning. For complex, i.e., composed concepts, the mere naming of real existing things or those of the imagination is sufficient to produce a lexical meaning.

In practice, this has its limitations, at least the same or similar naming, as Martin Fowler expresses in his article on the role pattern [32], which can lead to misunderstandings and misinterpretations. In the article *Dealing with Roles*, Fowler addresses the fact that just because a pattern is called a role pattern, this does not translate into the exact implementation and certainly does not intend the same semantics. However, if these implementations are conceptual, i.e., semantically identical, then the implementations can be substituted for each other. But he identifies about ten different patterns that are similar but not semantically identical. Furthermore, he lists the various advantages and disadvantages of these patterns, provided that it finally becomes apparent that a clear distinction of the pattern is necessary at the latest during the implementation.

What is the reason for the misinterpretation? – The cognitive scientist Lera Boroditsky investigated these phenomena and continued the theory about the relativity of language [33]. Boroditsky studied the influence of atomic concepts of a language on people's understanding and behavior in [34] [35]. She found in various experiments that different concepts of numbers, time, space, or of objects result in a generally different understanding of these concepts.

These and other experiments support the hypothesis that even the most basic concepts of human experiences, such as space, time, causality, and related objects, influence language and the nature of communication. The atomic concepts of language, its use, specific technical language, regional and even local differences, and the experience an individual has cause ways of thinking to be shaped by communication. This was investigated experimentally by selectively using metaphors that did not match the concepts of the language.

After multiple repetitions, corresponding changes in thinking styles were observed in non-linguistic implicit association tasks [36].

One challenge of software engineering is the duality between understanding programming as creative creation on the one hand and the structured approach on the other. Suppose this is viewed from the perspective of linguistics and cognitive science. In that case, this explains the choice of variable names and identifiers and the phenomenon observed by Dahl, Dijkstra, and Hoare. They describe in [37] that with experienced programmers, a programming style similar to a style with artists in a painting is to be recognized, and the higher the understanding of the structured programming is, the more clearly higher-level structures are formed.

The aspects considered so far are mainly reflected in the identifiers and the more abstract structures, wherein a similarity between formal and natural languages exists. What is the practical difference between a formal and a natural language, except for the complete mapping in the corresponding grammar? – Marcus Kracht investigated the emergence of syntactic structures [31] [38], for which he used concepts of programming languages and set theory in particular. His work is based on a well-founded theory, which he summarized in the monograph, *The Mathematics of Language* [39]. He also concludes that the following three features are underrepresented in any semantics of a natural language: Indexing, Multiplicity, and Order.

To pick out just one aspect, we take a look at the concept of indexing. The idea behind it is the well-ordering of a set. This exists according to Cantor's well-ordering theorem for any set [40]. Although the ordering is not always obvious, the distinction between two objects is essential if they are, in fact, different. This becomes clear when we look at the AST. Each use of a variable is linked to the declaration of that variable via the symbol table. In a specific program context (visibility), a variable name always designates the same variable. Therefore, a new name and an index must be assigned when a new variable is declared. So, in a "text" written in a formal language, we know about the identity of each object.

As essential for formalism, the meaning must be independent of the naming. By indexing and distinguishing multiplicities, as is necessary for programming languages, the semantics result from the fact that different things can be identified. Kracht's motivation for his work stems from understanding the notion of compositionality. For natural language texts, almost everyone has an intuition about which constructs are compositional and which are not. The mother tongue can be mentioned here as an example. Children do not use complex composition operators to determine which words should be combined in which way. Instead, they develop a sense for this over time, which results in an intuition for new compositions. However, these notions are based on something other than a formal foundation. He, therefore, calls for meaning to be defined without mention of syntax. From his point of view, it is not part of semantics to specify how things are composed in syntax [38]. This follows from the fact that the functionality of a program does not change if variables are named differently.

Kracht therefore defines a concept as described in Definition 4 as set $R_{\approx}$ of semantic equivalent relations. By a relation we understand in the mathematical sense a set of ordered pairs. We write if it holds that the relation R_1 and R_2 are semantically identical: $R_1 \approx R_2$

Two relations are considered semantically identical if they can be transformed into each other by the operations given in Definition 3. The operation OP_L-1 is also called type extension because a new element of the universe is added to the relation. On the other hand, with the operation OP_L-4, the relation is extended by an argument already contained. Even if after the operation always, only the last one is appended, the operation OP_L-3 can be applied before in such a way permuted as long as the desired argument stands in the last place.

Definition 4, relations and thus orderings are described, but the definition of the concept itself is independent of it and, therefore, its semantics. According to Kracht, a concept is defined by a set of relations. This means that the order is only reflected in the concrete realization, which corresponds to the intuitive understanding. It is also easy to regard an active and passive sentence construction as semantically equivalent, although the order is fundamentally different. On the semantic level, concepts thus describe a linkage. Concepts can therefore be regarded as equivalent, even if they use different relations for representation.

Kracht, in his article *Using Each Other's Words* [41] states, similar to Boroditsky's findings, that for any two people, even if they use syntactically exact words, they do not assign the same meaning to them. In summary, naming things, especially complex composed constructs, is difficult because the composed meaning is derived from how certain parts are put together and the individual meanings of those same parts. The meaning of the atomic parts is learned individually and is thus preassigned an individual understanding. Based on this learned understanding, emergent meaning is composed. In other words, a shared understanding of a complex concept can only emerge if a shared understanding of its parts exists. If there is none or if there is a different one, misunderstanding is inevitable.

C. The Description of Concepts through Examples

If we take realizable concepts to be sets of implementations of that concept, they can be described in two ways, intensional and extensional. We refer to the exemplars of these sets as Examples. Under the term architecture concept, we subsume architecture patterns as well as design and implementation patterns. In this paper, we concentrate on the extensional description of concepts. However, even if concepts cannot be described thoroughly, it is nevertheless the better choice, related to the use case of the detection and extraction of so far unknown concepts.

1) The Concept as a Set of Examples: The definition of the architecture concept described as a **named set** (see

Definition 4 (Concept (Linguistics), after ([31]p. 17 and [38], p. 65)): A concept is a set of relations $R_{\approx}$ with

$$R_{\approx} := \{R' \mid R \approx R'\} \tag{12}$$

Let Ω be the universe of a structure of first level predicate logic, then a relation R is a subset of Ω^n, $R \subseteq \Omega^n$. $R \approx R'$ then means that R' can be derived from R by any number of the following operations. Let $\vec{a} := (a_0, a_1, \ldots, a_n)$ and $\vec{a} \in R$, that is, an element from the relation R.

- (OP_L-1) Addition of an element from the basic set $R \mapsto R' \times \Omega = \{(\vec{a}, m) \mid \vec{a} \in P, \omega \in \Omega\}$
- (OP_L-2) Removing an element $R \times R' \mapsto R$
- (OP_L-3) Permuting the arguments $R \mapsto \pi(R)$
- (OP_L-4) Extension $R \mapsto \{(\vec{a}, a) \mid \vec{a} \in R\} = \{(a_0, a_1, \ldots, a_n, a_{n+1}) \mid (a_0, a_1, \ldots, a_n) \in R, a_{n+1} = a_n\}$

Definition 5 (Architecture Concept C (as named set)): Let CONCEPT be the universe of all known concepts. An architectural concept C is a named set of examples of the form

$$\begin{aligned} C &:= (id, R) \\ &\quad id \text{ a finite string for which an injective mapping } f \text{ exists,} \\ &\quad\quad \text{with } id \in \text{CONCEPT and } f : \text{CONCEPT} \to \mathbb{N} \\ &\quad\quad \text{it applies } \forall C_i, C_j, f(C_i) = f(C_j) \Rightarrow C_i = C_j \\ &\quad R \text{ a finite indexed set of semantically identical examples of the concept } C \\ &\quad\quad R := \{r_i \mid \forall r_j \in R \text{ it follows } r_i \overset{C}{\approx} r_j\} \end{aligned} \tag{13}$$

As short notation it is defined $C_{id} := R = \{r_i \mid \forall r_j \in R \text{ gilt } r_i \overset{c_{id}}{\approx} r_j\}$. The notation $r_i \overset{C_{id}}{\approx} r_j$ means that the two examples r_i ans r_j are semantically identical with respect to the concept C_{id}.

Definition 5), is done via the duple (id, R) and not directly by a typical set-notation, to account for the special meaning of the id, i.e., the name of the concept. By naming the set and choosing an identifier from the words of a natural language, this set, and thus the concept, acquires an induced meaning for humans. However, this simplicity is countered by the complexity and diversity of the descriptive examples, which are context dependent. This means that each example represents the concept within its own context. A (shared) understanding exists when the context-free identifier and the set of contextual examples have generated a context-independent understanding in the observer.

The context, therefore, reflects the area of application in which the concept is embedded. It describes a refinement and concretization of the concept within this context. At this point, a further principle of software engineering comes into action – the principle of domain orientation. With the application of a concept, a mixture and a fusion with elements of the domain inevitably occur.

Similar to a design pattern, a concept can be described by the triplet of problem, context, and solution. As introduced in the section before (Section V-B) the name is not a unique identifier for the equality of two concepts. In general, two sets are equal if they contain the same elements. Though, under what criteria are two elements, i.e., examples? – This depends on the way the examples are represented. At this point, therefore, we can only refer to semantic equality, but not to structural equality. The semantic equality of two concepts is given as. If C_1 and C_2 are each concepts, and if $R_{C_1} \cap R_{C_2} = \emptyset$ holds, which means that the set of examples of the two concepts is disjoint, then these two concepts are semantically identical, iff.

$$\forall r_i \in R_{C_1} \wedge \forall r_j \in R_{C_2} \mid r_i \overset{C_1}{\approx} r_j \tag{14}$$

This leads directly to the following equivalences:

$$r_i \overset{C_1}{\approx} r_j \Leftrightarrow r_j \overset{C_1}{\approx} r_i \Leftrightarrow r_i \overset{C_2}{\approx} r_j \Leftrightarrow C_1 \approx C_2 \Leftrightarrow R_{C_1} \cup R_{C_2}$$

It must be remarked that this form of equality cannot be used to make a statement about the structural equality of two examples of a concept. In linguistics, two concepts that are equal in this sense would be called synonymous.

Suppose the examples are concrete implementations, i.e., idioms, so programming language-dependent descriptions. In that case, the context consists of the specific language and the parts of the business logic contained in the respective example. In addition, for more complex concepts, the locality principle does not apply. Instead, the pattern, more precisely, the roles of the pattern, are distributed among different system artifacts. It follows that it is legitimate for concrete source code excerpts to appear in examples of different concepts.

2) Example Representation and Atomic Concepts: In Section III, an example of a concept – the Singelton pattern – was examined. As a representation, a graph representation based on the AST was introduced. In this case, it was used to provide concrete instances of an example. Since this representation is a semantically identical model transformation, no abstraction occurs.

But is the structure of a graph suitable to represent higher-level concepts? – Basically, such a structure is suitable to represent any objects and their relations among each other. Moreover, the set of elements (nodes) and the set of relations (edges) can be extended freely, including aspects that cannot be extracted directly from the source code. The subject of this work is the semantic relations, not the syntactic ones. This means that a back transformation into source code is not required. Therefore, any extension of the graph is also allowed at this point. In the previous section, linguistic concepts were assumed to be understood as a set of semantically equivalent relations. In principle, however, only two-digit relations are represented by edges in a graph. However, David Hilbert and

Wilhelm Ackermann found that any n-digit predicate can also be reduced to a less-digit one [42].

Generally, it is up to the modeling and thus the intended semantics of how the predicates are defined, thus also their rarity. This applies in this form only to the calculus of first-level predicate logic, but this is the object of investigation of Hilbert and Ackermann as well as of Kracht [31] [38]. A model based exclusively on triples, thus equivalent to a representation by a graph, is exemplified by the ontology [43].

We introduce the ASG to describe a more general formalism. In contrast to the concrete syntax tree, in which the nodes represent symbols of the grammar, the abstract syntax tree nodes represent concepts of the programming language and their hierarchical relationships. The structure is a simplification of the underlying grammar of the programming language. The edges represented in the AST are called syntactic edges E_{Syn} because they represent the syntax of the language. The edges constructed in the semantic analysis using the symbol table are called semantic edges E_{Sem}. By unifying these two sets of edges ($E_{\text{Syn}} \cup E_{\text{Sem}}$), the AST is extended from a tree to a directed graph, while it cannot be excluded that it is cycle-free.

The combination of syntactic and semantic edges is called Abstract Syntax Graph [44] and is specified in Definition 6. At this point, however, we must emphasize that this term is not a fixed term. The same applies to labeled AST, extended AST, or attributed AST, and it must also not be mixed up with the so-called term graph. Term graphs are often used in rewriting and automated refactoring and are often acyclic [45] [46].

The aggregated graph described in Section III-A and visualized in Figure 2 can be understood as a method-specific graph. The ASG here would be an intermediate representation between AST and aggregated graph. Compared to the ASG, the aggregated graph additionally has a node fusion operation performed. Like natural languages, atomic concepts exist for formal languages (cf. Definition 3). With respect to the representation of ASG, the notion of atomic concept is extended as follows:

Definition 7 (Atomic Concept): An atomic concept is a concept defined by the syntax, i.e., by terminal or non-terminal symbols of the programming language grammar, and represented by a node or edge type in ASG.

3) The Minimal Example: In this section, the reduction aspect of an example is discussed. From the definition for graphs, the following can be said in general about the features of a graph g.

1) It is directed,
2) can be a multi-graph (A multi-graph is a graph with more than one edge between two nodes),
3) has in general cycles (Cycle-free minimal examples exist, but they are trivial examples of atomic concepts) and
4) is connected.

In the previous sections, any graph was considered an example of a concept. It was only required that in the graph, at least once in the underlying program code, the named concept was applied. If several examples of a concept are compared, these may have a different number of nodes and edges. Therefore, it is not only possible to determine a minimal example among the existing ones, but a minimal example exists for each example. To illustrate this, the atomic concept *Class* is considered. If a method is added to a class, it remains an example of a class; even if this is repeated several times, the truth of the statement does not change. Even a class that has no methods can be considered an example of a class. This example illustrates two perspectives. From the context-sensitive view of the respective example, it can be examined which elements are not essential for the concept. From the context-independent view of the concept, the requirements for the example can be defined. If, for example, the concept *ClassWithMethod* is described, we expect at least one class with one method. The so-called minimum concept can be described as follows.

Definition 8 (Minimal Example of an Architectural Concept):

A minimal example of an architectural concept is an example, i.e., a piece of program code for which holds: if any character is removed from this piece of program code, then this is no longer an element of the set of examples of the respective concept.

What results for the ASG of a minimal example? – For an atomic concept, as in the example of the Class, this is defined by the programming language, namely by the minimal syntactically correct root tree where the root is of type class declaration. In this concrete case, it depends on the parser and grammar, different designations of the type are very likely. For non-atomic concepts, i.e., those that satisfy the compositionality principle, structures must exist that allow this concept to be decomposed. In the context of design patterns, which are widely accepted as architectural concepts, proven description templates have been discussed, as exemplified in [2] and [47].

One aspect of these description templates are the participants, also roles, of which the pattern is composed. Complex composed concepts are thus ascribed the existence of roles.

*Definition 9 (**Role** (according to [48])): A role is a observable behavioral aspect of an object in a concrete context.*

For a graph $g(V, E)$ representing an example of the concept C, it follows from conclusion 2 that there can be nodes that are not elements of a role, and thus not part of a behavioral aspect of the concept.

If we consider the collaborations of the UML as a description technique, then the parts, which are named collaboration roles, serve as a structural element [49]. In addition, other elements in UML are called roles in simple association, for example, as in the class diagram. In Figure 11, two classes and a binary association are shown. In such an association, the association ends are called roles. Here, the name of the association describes the semantics of the association, and the name of the role describes the meaning of the class related to this association [50]. We claim that a formalism should be independent of naming. This holds also for the naming of associations if we understand them as concepts. Moreover, we

Definition 6 (Abstract Syntax Graph $g_{ASG}\left(t_{AST}(V_{Syn}, E_{Syn}), E_{Sem}, P, \tau\right)$): Let $t_{\text{AST}}(V_{\text{Syn}}, E_{\text{Syn}})$ be the AST (see Section III) of a program, then the abstract syntax graph is the extension corresponding to a tuple with the following signature:
$g_{ASG}(V, E) := \left(t_{\text{AST}}(V_{\text{Syn}}, E_{\text{yn}}), E_{\text{Sem}}, P, \tau\right)$

$$
\begin{array}{ll}
V_{\text{Syn}} & \text{a finite } \textbf{Set of nodes}. \\
E_{\text{Syn}} & \text{a set of } \textbf{syntactic edges}. \\
E_{\text{Sem}} & \text{a set of } \textbf{semantic edges}. \\
P & \text{a finite set of } \textbf{edge-terminals} \\
\tau & \text{a relation that maps each edge from } E_{\text{sem}} \text{ maps to an edge terminal.} \\
 & \tau := E_{em} \rightarrow P \\
 & \forall e \in E_{\text{Sem}} \mid !\exists p \in P \wedge \tau(e) = p
\end{array}
\tag{15}
$$

It holds that the node set of the ASG is equal to that of the AST:

$$V := V_{\text{Syn}} = \Lambda \cup T \cup \Sigma \tag{16}$$

Only the set of edges is extended, so that the following is true:

$$E := E_{\text{Syn}} \cup E_{\text{em}} = \gamma \cup \omega \cup \tau \tag{17}$$

speak in this case about associations, which are an object of subjective consideration since they no longer belong to the atomic concepts of the language; instead, they are composed concepts.

Fig. 11: Elements of a binary association according to [49], p. 135

Transferred to concepts according to Definition 5 results:

*Conclusion 1 (**Roles in Concepts**):*

A concept consists of a finite non-empty set of roles manifested in the examples or an atomic concept.

This conclusion is consistent with the role pattern, a widely used pattern of object-oriented analysis (OOA) [50] [51] [32]. A description of this pattern is given in the Figure 12. Three main features emerge from this description.

1) The item is role independent. In [47] this is also called the core class. It contains static, immutable functionality.
2) The item can take on different roles in any number of contexts.
3) The assignment can change dynamically and the item thereby aggregates the contexts to itself and thus dynamically receives its role-dependent properties and functionalities via the roles, which are described as association classes.

At this point, the variant Role-Relationship according to [32] was selected on purpose. According to the same paper, this variant is suggested if an item-object can take more than one role concerning another object, as is the case in this work if the system is chosen as context. However, this can also be true for an example of a concept. In the further course, this notion of role is further restricted to minimal examples.

The choice of modeling the role as an association class is also made on purpose. Other variants and especially perspectives in which the context is not explicitly modeled are given in [47] and [50]. The association class clarifies the membership of role-specific attributes and operations. It describes

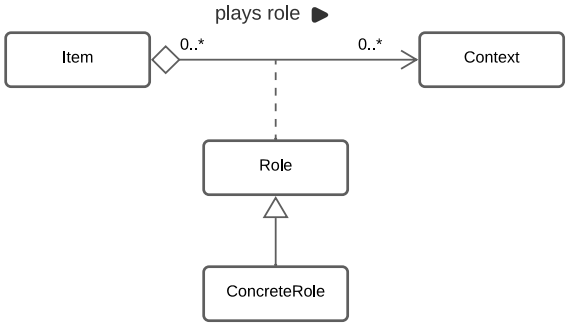


Fig. 12: The role pattern as UML class diagram and modeled as association class

properties that result from the relationship between the object and the context and cannot be meaningfully assigned to either class. The property of whether an object fills a particular role is no longer well-defined because the same class can occupy different roles in different contexts. The application of the role pattern destroys the well-definedness of the concept.

*Conclusion 2 (**Roles as Subgraphs**):*

With respect to the chosen representation, roles manifest themselves in subgraphs of a graph g of the conceptual example. Let h_1 and h_2 be roles, thus two real subgraphs of g, $h_1, h_2 \subset g$, then the following statements follow from the definition of role (Definition 9): (i)

1) $h_1, h_2 \subset g \nRightarrow h_1 \cap h_2 = \emptyset$
2) $g \setminus g' \neq \emptyset$ mit $g' := \bigcap_{h_i \subset g}$

Roles do not define an equivalence relation on a concept.

In general, a complex concept consists of more than one behavioral aspect; at this particular step, only their existence is assured. The concrete roles are assumed to be unknown in the paper if not stated otherwise.

Therefore, for a minimal example, each node of the associated graph belongs to at least one role. All other nodes can therefore be removed from the graph. Furthermore, all nodes not essential for the concept can be removed; the same applies to the edges.

This results in a minimal example of architectural concept defined in Definition 10.

*Conclusion 3 (**Connection of minimal Examples**):*

However, deletion operations may cause a minimal example to no longer be connected. For an example, since it is constructed from the ASG, it follows that it is connected. There may be dependencies which only arise dynamically at runtime, if such a link is the only one connecting two nodes, then this link is a bridge in the sense of graph theory and the graph decomposes into more than one component.

Another case represents the removal of the root element when the AST is considered as the spanning tree of the ASG. In many compilers, a parent node describing the combining unit is introduced as the root node, and in most cases, this is not essential to the concept and can therefore be removed.

In addition to removing, adding a node or an edge in an example is possible, also. The above example with the atomic concept of the Class could lead to the impression that these operations are possible without any problems. This is not true in general. For example, there are concepts where these operations can be repeated any number of times for particular nodes or edges, but there are also those for which this is not true. In the later part of the paper, the singleton pattern will be examined. If a public constructor is added to some variants of this pattern, it is no longer a valid example of this concept. This addition has destroyed the concept.

However, by introducing the minimal example, the instance term can be defined.

Definition 11 (Instance of a Concept):

Let $S \in$ SYSTEM be any system, g the associated ASG, then the subgraph $h \subseteq g$ is an instance of the concept $C(id, R)$ iff. h is a minimal example of C, so $h \in \hat{R}$ holds.

This also corresponds to the instance concept of object orientation. The correct formulation for a non-minimal example is that it contains an instance and does not represent it directly.

If an architectural concept C contains only minimal examples in its set of examples, then this is called a minimal architectural concept $\hat{C}$ as described in Definition 12.

Furthermore, it follows directly from Definitions 5 and 12 for architecture concepts:

$$\hat{C}_{id} \subset C_{id} \mid id = id \tag{19}$$

This means that there is a real subset relationship between the set of examples of an architectural concept and the set of examples of the associated minimal example. The size of the set of examples of a concept is countably infinite. A real subset of this set is formed by the minimal examples. Also for natural language concepts Kracht describes, relations which are minimal with respect to their length and thus in a certain sense even special. If $\hat{R}_i$ and $\hat{R}_j$ are two minimal relations of a linguistic concept (Definition 4), then $\hat{R}_j$ is a permutation of $\hat{R}_i$ and it holds [38]:

$$\pi(\hat{R}_j) = \hat{R}_i \tag{20}$$

Applied to architectural concepts, this would mean that two minimal examples $\hat{r}_i$ and $\hat{r}_j$ of a concept are isomorphic to each other, $\hat{r}_i \cong \hat{r}_j$. As a result, not only would there be a bijective mapping between these graphs, but the graph invariants, such as the number of edges or nodes, should not change. Nevertheless, this cannot be guaranteed in general. This is caused by operations such as fusion and contraction. As a result, the problem can be written as a partial graphisomophy problem Subgraph-Isomorphism (SGI) [52] [17], which is assigned to the complexity class of NP-complete problems. These aspects are discussed in more detail in the following chapter, where identification and extraction use cases are examined.

VI. Related Work

A similar approach to the one we propose is code2vec [53] [54], also working with an abstraction based on a set of paths. The main difference is the structure of the extracted path. All pairwise paths between the leaf nodes are examined and limited to a maximum number and length. They define the path-context by a triplet $< x_s, p, x_t >$, where x_s is the start leaf, x_t is the target leaf, and p the path between these nodes with the additional information whether a traversal takes place upwards towards the root element or downwards in the tree. The approach is presented here all paths from each leaf to the root are taken into account. Another limitation of code2vec is the abstraction context, which is one method. They argue that the order of source code statements is not relevant, or valid for this scope and the defined task. But as shown in [55], the relation between source code elements for higher concepts (like classes) is essential to perform structural or behavioral related tasks. As shown in [56] another limitation of code2vec is its sensitivity to naming. For tasks like those described in code2vec, where names of methods are predicted, names are of course essential, but for the extraction of abstract concepts, the uncertainty of the correct name is too high.

Yarahmadi et al. [55] have conducted an extensive and systematic literature review on how design patterns can be detected in code and therefore abstract the code to perform this task. The main findings of this study relevant to this paper are: Many of the approaches have been tested and evaluated only on small data sets or on limited code samples. The principle in almost all approaches that were reviewed is to reduce the search space by abstraction. Most approaches were limited in their ability to recognize different types of patterns. Another problem of many approaches is detecting different variants of a pattern. To make this possible, ML methods are often used. However, these methods require good data preprocessing because it is not possible to decide in a general way which parts should be selected for learning. A common approach to this problem is, as implemented in [57], a semi-automatic approach in which a human takes over feedback or labeling.

Another principle often used in addition to using the syntactic concepts of programming languages is to analyze the identifiers (e.g., classes, methods, or variable names) using natural language processing techniques [58] [59]. Schindler et al. [59] demonstrated that these methods are well suited for project-specific domain models but not for identifying general

Definition 10 (Minimum Example $\hat{r}$ (formal)): Let C be a concept and r an example in which it is applied, then this is called minimal, written $\hat{r}$,iff. $\hat{r}$ is a minimal subgraph of r satisfying the following conditions. (i)

1) $g \setminus g' = \emptyset$ with $g' := \bigcap_{h_i \subset g}$ with $H := \{h_1, \ldots, h_n\}$ the set of all roles of C
2) $\forall v \in V_{\hat{r}} \mid V_{\hat{r}} \setminus \{v\} \Rightarrow \hat{r} \notin R_c$
3) $\forall e \in E_{\hat{r}} \mid E_{\hat{r}} \setminus \{e\} \Rightarrow \hat{r} \notin R_c$
4) $\forall v \in V_{\hat{r}},\ \forall m \in \mathbb{N} \mid 1 \leq m \leq \mid \mathfrak{P}_c \mid \wedge\ f_v(v) - (\vec{\mathfrak{p}}_{i,j})_m \Rightarrow \hat{r} \notin R_c$
5) $\forall e \in E_{\hat{r}},\ \forall m \in \mathbb{N} \mid 1 \leq m \leq \mid \mathfrak{P}_c \mid \wedge\ f_e(e) - (\vec{\mathfrak{p}}_{i,j})_m \Rightarrow \hat{r} \notin R_c$

Definition 12 (Architecture Concept $\hat{C}$ (minimal)):
Let CONCEPT be the universe of all known concepts. By Definition 5, an architecture concept C is a named set of examples. If all examples of this set are minimal, then this is denoted by the notation $\hat{C}$ and is defined as:

$$
\begin{aligned}
&\hat{C} := (id, \hat{R}) \\
&\quad id \text{ a finite string for which an injective mapping } f \text{ exists,} \\
&\quad\quad \text{with } id \in \text{CONCEPT and } f : \text{CONCEPT} \to \mathbb{N} \\
&\quad\quad \text{it holds } \forall C_i, C_j, f(C_i) = f(C_j) \Rightarrow C_i = C_j \\
&\quad \hat{R} \text{ a finite indexed set of semantically identical minimal examples with respect to the concept } C. \\
&\quad\quad \hat{R} := \{\hat{r}_i \mid \forall \hat{r}_j \in \hat{R} \text{ gilt } \hat{r}_i \overset{C}{\approx} \hat{r}_j \wedge \hat{r}_i \text{ is minimal}\}
\end{aligned}
\tag{18}
$$

patterns. Natural language identifiers can be an indication but not a robust criterion. An example of how the AST is able to be enriched by additional features, e.g., by using ML, is described in [60] and [61].

In addition, tools and frameworks should also be mentioned, which could also be applied, though in part with restrictions. For example, jQAssistent [62] is a tool that transfers the AST into a Neo4j graph database, offers the possibility of manually enriching this graph with further information, and then using the query-language Cypher to define concepts and identify them in the graph. In contrast to the approach presented in this paper, a query needs to be formulated covering the concept for which the sample should be retrieved.

ArchUnit [63], Structure101 [64], and Dependometer [65] are based on the same principle of formulating rules that are checked automatically afterward. However, the creation and management of rules is costly with the increasing complexity of the concept, and require substantial expert knowledge. All of the mentioned approaches do not assist in expressing rules applying to a given set of samples.

The major problem in this kind of approach and any other approach based on a specific formal language is that it is difficult to define the concrete rules describing a pattern correctly. Rasool et al. [66] describe it as a lack of standard specification for design patterns.

The field of code clone detection is related to the approach presented in this paper since the input data is identical. In [67], four types of code clone detection are characterized, (i) syntactically identical code fragments, (ii) syntactically identical except names and literal values, (iii) syntactically similar fragments that differ in some statements but can be transformed to each other by simple operations and (iv) syntactically dissimilar code fragments but sharing the same functionality. In contrast to code clone detection, we do neither want to find syntactically identical fragments (i)-(iii) nor functionally identical ones (iv). Because of the domain-specific adaptation, we are not interested in finding direct copies.

VII. CHALLENGES OF EXTRACTING ARCHITECTURAL CONCEPTS

In Section V, a formal description of general concepts was derived based on set theory, graph theory, and linguistics. This description can only be understood as part of a higher-level methodical approach, described in, for example, in [61] [68] [69]. In this context, we are confronted with practical application challenges which arise in the exchange between programmer and architect. The cause of this tension is based on the different perspectives on the system, i.e., the architecture description on the one hand and the implementation on the other hand. The following four aspects are examples of this: (I.) *How can a **continuous mapping** between an architectural concept as part of an architectural description and its implementation be created?* (II.) *How to ensure **semantic correctness** between concepts in the architectural description and concepts in the implementation?* (III.) *How can the **completeness** of the concepts in the architecture documentation be ensured with respect to the concepts implemented in the implementation?* (IV.) *How can the maximum possible conceptual knowledge be extracted from **minimal data**?*

A. Continuous Mapping

Due to the selection of the considered challenges, marking arbitrary concepts in the ASGs is necessary. The highlighting must be possible so that different instances of the same concept and parts of the concept that do not manifest themselves according to the locality principle can be mapped, and the compositionality of concepts is considered.

A metaphor introduced for this purpose is the concept of **Color**. In graph theory, the concept of coloration is used for various questions [70]. Nevertheless, Color is also suitable as a metaphor independently of any mathematical structure since aspects such as nuances in the form of brightness and saturation, or even the creation of new colors by mixing primary colors, for example, can be easily imagined. Moreover, it can be applied to the composition of concepts. Color highlighting is also a very well-understood concept in different contexts.

Definition 13 (Color $\mathfrak{p}$): Let COLOR *be the universe of all known colors. A color* $\mathfrak{p} \in$ COLOR *is a label which a node or an edge of a graph may possess. A color is uniquely identified by its name, that is, there exists an injective mapping* f : COLOR $\rightarrow \mathbb{N}$*, where holds.*

$$\forall \mathfrak{p}_1, \mathfrak{p}_2 \in \text{COLOR} \mid f(\mathfrak{p}_1) = f(\mathfrak{p}_2) \Rightarrow \mathfrak{p}_1 = \mathfrak{p}_2 \tag{21}$$

If colors are to be used to mark both atomic and complex concepts, then it is easy to see that not only nodes and edges must be marked, but also entire subgraphs must be able to be assigned a color. In order to allow multiple applications of a concept as well, a mechanism of instantiation of colors concerning a concrete graph must be introduced. A corresponding indexing of colors in the following implements this.

Let g be a graph and $\mathfrak{P} \subseteq$ COLOR where $\mathfrak{P} := \{\mathfrak{p}_1, \mathfrak{p}_2, \mathfrak{p}_3\}$ is the finite set of colors to be used for marking in the graph and

$$\mathfrak{P}_g := \mathfrak{p}_{1,1}, \mathfrak{p}_{1,2}, \mathfrak{p}_{1,3}, \mathfrak{p}_{2,1}\mathfrak{p}_{2,2}, \mathfrak{p}_{3,1}\}$$

the concrete instances of the colors.

Conclusion 4 (System Boundary of Instances): The graph is a system boundary with regard to the instances of a color. Given two color instances $\mathfrak{p}_{i,n} \in \mathfrak{P}_g$ *and* $\mathfrak{p}_{j,m} \in \mathfrak{P}_h$*, then holds:* (i)

1) $\mathfrak{p}_i, \mathfrak{p}_j \in \text{COLOR} \mid \mathfrak{p}_i = \mathfrak{p}_j \Rightarrow i = j$
2) $\mathfrak{p}_{i,n} \in \mathfrak{P}_g,\ \mathfrak{p}_{j,m} \in \mathfrak{P}_h \mid i = j \wedge n = m \nRightarrow \mathfrak{p}_{i,n} = \mathfrak{p}_{j,m}$

As a result of the described gap between implementation and architecture, there is no direct injective mapping between these two artifacts. The implementation always represents a contextual application of architectural concepts. This makes setting up this mapping in particular difficult. Related to the coloring of the abstract syntax graph, continuous mapping exists precisely when all nodes and edges belonging to an instance of a concept are marked. Of course, this also assumes that if the example is according to Definition 5, all aspects of the concept have already been fully implemented in the implementation. Otherwise, it is possible to speak of a marking of the concept but not of an example.

The labeling principle is formulated in such a general way that even new concepts can be mapped intuitively in the sense of previously unknown. According to the definition of an architectural concept (Definition 5) and that of a color (Definition 13), a bijective mapping can be formulated between concepts and colors. In the form that the concept which is mapped to $n \in \mathbb{N}$ is represented by the color, which is also mapped to n. Defining a new concept is equivalent to creating a new color. However, what does it mean that a new color/concept is created? – Two cases must be distinguished: (i.) introduction of a new atomic concept and (ii.) introduction of a new composed concept.

The universe of colors is extended in both cases. One difference is that in case (i.) an extension of the underlying programming language takes place and is a cross-system concern. This is not true for case (ii.). The trigger for case (ii.) consists of the fact that a concept is to be selected, which is not yet named in the form. Accordingly, there is no color and no example yet for this concept. This labeling of a concept instance has the character of an annotation and is initially only valid for the system under consideration. Thus the ASG is extended by information that cannot be derived directly from the program code. For this reason, the so-called **Concept-Graph** is introduced for a conceptual separation. We define that an abstract syntax graph (Definition 6), which is colored accordingly, is called **Concept-Graph**. This graph may furthermore be extended by so-called **Concept-Nodes** $V_{Concept}$ / **Concept-Edges** $E_{Concept}$. A concept graph is thus defined as follows.

Definition 14 (Concept-Graph): Let $g_{ASG}(V, E)$ *be an abstract syntax graph of a program with node set* $V = V_{Syn}$ *of AST and edge set* $E := E_{Syn} \cup E_{Sem}$ *of ASG). A colored directed Concept-Graph* $g \in$ GRAPH *is represented by the following signature:* $g(g_{ASG}, \mathfrak{P}_{\text{ATOM}}, V_{Concept}, E_{Concept}, \mathfrak{P}_{Concept}, f_V, f_E)$

$$\begin{aligned}
&V \text{ a finite indexed set of nodes.}\\
&\quad V = V_{ASG} \cup V_{Concept}, V := \{v_1, v_2, \ldots, v_n\}\\
&E \text{ a finite indexed set of directed edges}\\
&\quad E \subseteq V \times V,\ E := \{e_1, e_2, \ldots, e_m\} \text{ mit } e_i = (v_j, v_k)\\
&\quad E = E_{ASG} \cup E_{Concept}\\
&\mathfrak{P} \text{ a finite indexed set of color instances}\\
&\quad \mathfrak{P} = \mathfrak{P}_{\text{ATOM}} \cup \mathfrak{P}_{Concept}\\
&\quad \mathfrak{P} := \{\mathfrak{p}_{1,1}, \mathfrak{p}_{1,2}, \ldots, \mathfrak{p}_{1,i}, \ldots, \mathfrak{p}_{n,1}, \ldots, \mathfrak{p}_{n,j}\}\\
&\quad \text{mit } \{\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_n\} \subseteq \text{COLOR}\\
&f_V \text{ Coloring function for nodes } f_V : V \rightarrow \mathbb{Z}_2^{|\mathfrak{P}|}\\
&f_E \text{ Coloring function for edges } f_E : V \rightarrow \mathbb{Z}_2^{|\mathfrak{P}|}
\end{aligned} \tag{22}$$

Even if the node types are represented by colors concerning the programming language's grammar, in general, we should no longer speak of typed nodes and edges, only of colored ones, because it is now possible to assign several colors to each node or edge. However, this is contrary to the definition of the type. On this basis, now arbitrary architecture concepts can be represented, and further operations on these can be defined.

B. Semantic Correctness

Suppose a named set of examples exists, validated in that the contained examples are semantically equivalent. In that case, it holds that the given examples are context-independent, as well as an interpretation of the name in the form of these examples is possible. Nevertheless, no complete description of the concept exists in the form of a specification. Furthermore, the examples are semantically equivalent but structurally different. This is caused by the embedding in an application context. In the Limitations (see Section IV-C), the maximum common subgraph problem has already been pointed out. One is confronted with the same NP-hard problem if we want to answer the question of whether a graph g should be included

as an example in the set of examples of a concrete concept $C(id, R)$?

$$g \in R \text{ or } g \notin R$$

We can define the concept of a **Dector** as follows:

Definition 15 (Detector d_C): A detector $d_C \in$ DETECTOR classifies a graph g if the graph is an example of a given concept $C(id, R)$ or not.

$$d_C : g \rightarrow [\text{true},\ \text{false}]$$

$$d_{c_i}(G) := \begin{cases} \text{true}, g \in R \\ \text{false}, g \notin R \end{cases}$$

The goal is to develop a methodology to identify known concepts (identified by identifiers and described by examples) in a given program using a heuristic to derive suggestions for new concepts from the extracted information. Since this is an NP-hard or partly an NP-complete problem, it is reasonable to modify the concept of semantic correctness so that we can refer to a semantic similarity instead. The consequence of this is that there must be an expert who takes over the validation related to the correctness. However, this can also be an advantage, as it makes it possible to create specific concept variants, including project or domain-specific implementations of a concept.

A corresponding adaptation can be achieved by fuzzifying the edges to hyperedges and the coloring functions of the concept graph. If these are fuzzified, then the binary vector becomes real in the interval 0 to 1. In this case, it is a so-called **Fuzzy-Hypergraph** as described in Definition 16 and would have to be considered accordingly in the example representation.

The idea behind the Fuzzy-Hypergraph is that nodes can be grouped with a given membership function σ, μ, and a given degree α. Each hyperedge here represents exactly one aspect or parts of a hierarchical aspect. This allows us to express for a concrete instance which influences specific nodes have in this context. Fuzzy-Hypergraphs are suitable for partitioning tasks and pattern recognition, among other things. Since this is an analytical approach, the quality of the results is strongly dependent on the modeling of the membership functions.

C. Complettness of Extraction

The documentation of a system's architecture is always an incomplete system description. This is, on the one hand, because only those aspects are listed which have relevance from the point of view of the architect and, on the other hand, due to the fact that programmers establish concepts during their development but do not communicate this. The completeness concept is dependent on the extraction of subjective feeling. We can say, therefore: The list of the implemented concepts is complete if the architect sees all the concepts known to him or concepts, which are already present in the knowledge base, are not proposed anymore and validated as positive. If other concepts are present in the implementation, and they will be, then they are not significant at the current time or dedicated as such.

The more interesting case, however, is the one where the programmer implements concepts that are not known to the architect. A case in extracting new concepts is when the previous knowledge consists only of atomic concepts. Here, no other coloring can be performed. Finding new concepts (colors) for nodes and edges can be traced to a clustering problem. Outlier detection, i.e., frequencies, outliers, or neighboring / overlapping graphs, must be searched for. Here, methods can be used as described in Section III-A. Given the formalization, it can be assumed that different methods must be combined for different aspects.

Having a procedure that is open to new concepts means that an integration mechanism must exist to incorporate them into a knowledge base. For example, higher-level concepts are formed from the programming language's atomic language concepts (colors) and their relationships to each other. In other words, relationships exist between the colors, and precisely these relationships must now be extracted. Thus also, the integration mechanism can be specified in the introduction of new colors as well as the way of assigning colors to nodes, edges, and graphs. This poses particular challenges to the methodology, not only for the one-time extraction but also for the evolution of the system under investigation over time.

This question highlights once again that the architecture of a system cannot be considered invariant, even if an architecture should rather be stable over the system's life cycle and represent a default for the implementation [72]. During implementation, situations may arise that require a change of concept. But such an architectural decision must be made consciously for all and ideally documented and justified.

D. Data Challenge

The more complex a concept grows, the fewer examples exist for this concept; as an example for the evaluation of the presented method (see Section III-A and IV), the Singelton pattern was used. With this pattern, it was possible to build up a corresponding dataset. However, this had to be validated manually because in the result set, despite the name Singelton pattern, there were examples that were not semantically equivalent to the chosen specification. The reason for this is explained in detail in Section V-B. Therefore, expert checking should always be considered.

Considering patterns that are more system-wide concepts, such as Pipes-And-Filters [73] or a Layered architecture, it is common for this concept to manifest itself only once in the system. This means that hundreds of systems with this corresponding architecture are needed for a similar number of examples as the extraction was performed for the Singelton pattern. On the other hand, the concept behind layered architecture, for example, can be described in a fraction of the size of the examples.

As a result, methods must be constructed that can work with a small amount of data, address the compositional aspect of a concept, and construct results that humans can easily

Definition 16 (Fuzzy-Hypergraph $\widetilde{\widetilde{h}}$ $(V, \mathcal{E}, \alpha, \sigma, \mu)$, see [71]): A Fuzzy-Hypergraph $\widetilde{\widetilde{h}}$ is represented by the following signature:

$$
\begin{aligned}
&\widetilde{\widetilde{h}}\ (V, \mathcal{E}, \alpha, \sigma, \mu) \\
&\qquad V \text{ a finite indexed set of nodes, } V := \{v_1, v_2, \ldots, v_n\} \\
&\qquad \mathcal{E} \text{ a finite indexed set of Hyper-Edges. } \mathcal{E} \subset \mathcal{P}(V) \setminus \emptyset \\
&\qquad\quad \mathcal{E} := \{\tilde{E}_1, \tilde{E}_2, \ldots, \tilde{E}_n\} \\
&\qquad \sigma \text{ a Fuzzy-Set } V_\alpha,\ \sigma : V \to [0, 1] \\
&\qquad\quad V_\alpha := \{v_i \in V \mid \sigma(v_i) \geq \alpha \wedge \alpha \neq 0\} \\
&\qquad \mu \text{ a set of membership functions } \mu := \{\mu_1, \mu_2, \ldots, \mu_{|\mathcal{E}|}\} \\
&\qquad \tilde{E}_i = \{v_j \in V \mid (v_j, \mu_i(v_j)) \geq \alpha \wedge \alpha \neq 0\} \\
&\qquad \bigcup_{i=1}^{|\mathcal{E}|} supp(\tilde{E}_i) = V
\end{aligned}
\tag{23}
$$

Where $supp(A)$ is the support of the set and is defined as:

$$
supp(A) := \{x \in A \mid \mu_A(x) \geq 0\} \tag{24}
$$

interpret. For example, the approach described in this article. Moreover, a possible approach could be to investigate logic-based representations as described in Herold [74] and Deiters [21] [22]. In this way, an approach could be developed based on extensional and intensional concept descriptions.

VIII. Conclusion and Future Work

Starting from the approach, which was introduced in [1], an extension was made in this article. We derived a comprehensive theory and formalism, which makes it possible to establish holistic approaches as they are described in Herold et al. [75], Knieke et al. [68] and Schindler [61]. All of these approaches try to mitigate architecture degradation using ML. Focusing on extracting concepts from existing implementations, most common approaches like code2vec, for example, rely on large amounts of data, so they are unsuitable for this kind of problem. Because, on the one hand, these data have to be acquired and validated, and on the other hand, the results have to be interpretable by humans.

We have shown an approach to extract the essence of a shared concept driven by available implementations so that the formulation is interpretable by humans. Moreover, we formulate expressions that explicitly be not part of an implementation of the concept. In other words, if such a path were added to any concept example, it would destroy it.

Future work planned includes two directions, on the one hand addressing the way the semantics of a concept is described and, on the other hand, using the introduced representation and abstraction technique as a preprocessing step in the direction of ML techniques. For example, to train a classifier or cluster samples to identify variants or the inner parts of a pattern, e.g., roles. Including the addressed limitations and collecting a high quality and high quantity data set of different design patterns, including different variants of a pattern. We choose an extensional description for the semantics. Experiments have shown that if a set of examples of a concept is known and validated, describing them intensionally using predicate logic formulas could be possible. The question of describing and interpreting a composition operator for architectural concepts can be seen as essential and still open at the current research stage.

References

[1] C. Schindler, M. Schindler, and A. Rausch, "Negligible details - towards abstracting source code to distill the essence of concepts," in *ADAPTIVE 2022*, M. Kurz, Ed. Wilmington, DE, USA: IARIA, 2022, pp. 22–31. [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=adaptive_2022_2_20_50009

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: Elements of reusable object-oriented software*, 2nd ed., ser. Addison-Wesley professional computing series. Boston: Addison-Wesley, 1997.

[3] J. Coplien, *Software Patterns*. SIGS Books & Multimedia, 1996.

[4] K. Bergner, A. Rausch, and M. Sihling, *Using UML for Modeling a Distributed Java Application*. TUM, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.6797

[5] G. Sunyé, A. Le Guennec, and J.-M. Jézéquel, "Design patterns application in uml," in *European Conference on Object-Oriented Programming*, 2000, pp. 44–62.

[6] S. Hussain, J. Keung, and A. A. Khan, "The effect of gang-of-four design patterns usage on design quality attributes," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017, pp. 263–273.

[7] C. Deiters and A. Rausch, "Assuring architectural properties during compositional architecture design," in *International Conference on Software Composition*. Springer, 2011, pp. 141–148.

[8] M. Paixao, J. Krinke, D. Han, C. Ragkhitwetsagul, and M. Harman, "Are developers aware of the architectural impact of their changes?" in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 95–105.

[9] M. Schindler and S. Lawrenz, "Community-driven design in software engineering," in *Proceedings of the 19th International Conference on Software Engineering Research & Practice, Las Vegas, NV, USA*, 2021.

[10] M. L. Scott, *Programming language pragmatics*, 4th ed. Amsterdam and Boston and Heidelberg and London and New York and Oxford and Paris and San Diego and San Francisco and Singapore and Sydney and Tokyo: Morgan Kaufmann/Elsevier, 2016.

[11] N. Chomsky and D. Lightfoot, *Syntactic structures*, 2nd ed., ser. A Mouton classic. Berlin: Mouton de Gruyter, 2002.

[12] N. Chomsky, "Three models for the description of language," *IEEE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[13] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, techniques, & tools*, 2nd ed. Boston: Pearson Addison Wesley, 2007.

[14] J. Niere, J. P. Wadsack, and L. Wendehals, "Handling large search space in pattern-based reverse engineering," in *11th IEEE International Workshop on Program Comprehension, 2003.* IEEE, 2003, pp. 274–279.

[15] A. Rajaraman and J. D. Ullman, "Data mining," in *Mining of Massive Datasets*, A. Rajaraman and J. D. Ullman, Eds. Cambridge: Cambridge University Press, 2011, pp. 1–17.

[16] P-mart pattern-like micro-architecture repository. [retrieved: 03, 2023]. [Online]. Available: https://www.ptidej.net/tools/designpatterns/index_html

[17] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, ser. A series of books in the mathematical sciences. New York u.a: Freeman, 1979.

[18] V. Kann, "On the approximability of the maximum common subgraph problem," in *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 1992, pp. 375–388.

[19] "Concept," 2022, [retrieved: 03, 2023]. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/concept?q=concept

[20] E. Zermelo, "Über grenzzahlen und mengenbereiche," *Fundamenta Mathematicae*, vol. 16, no. 1, pp. 29–47, 1930. [Online]. Available: https://eudml.org/doc/212506

[21] C. Deiters and A. Rausch, "Assuring architectural properties during compositional architecture design," in *Software composition*, ser. Lecture Notes in Computer Science / Programming and Software Engineering, S. Apel, Ed. Berlin and Heidelberg: Springer, 2011, vol. 6708, pp. 141–148.

[22] C. Deiters, *Beschreibung und konsistente Komposition von Bausteinen für den Architekturentwurf von Softwaresystemen*, 1st ed., ser. SSE-Dissertation. München: Dr. Hut, 2015, vol. 11.

[23] H.-D. Ebbinghaus, *Einführung in die Mengenlehre*, 5th ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. [Online]. Available: http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1878742

[24] B. Russell, *The philosophy of logical atomism*, ser. Routledge Classics. Abingdon, Oxon: Routledge, 2009. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10330922

[25] D. Gutzmann, *Semantik: Eine Einführung*, ser. Lehrbuch J.B. Metzler. Berlin and Heidelberg: J.B. Metzler Verlag, 2020.

[26] W. Hodges, "Formalizing the relationship between meaning and syntax," in *The Oxford handbook of compositionality*, ser. Oxford handbooks in linguistics, M. Werning, W. Hinzen, and E. Machery, Eds. Oxford: Oxford Univ. Press, 2012.

[27] D. Hillert, Ed., *Die Natur der Sprache: Evolution, Paradigmen und Schaltkreise*. Wiesbaden: Springer, 2017.

[28] J. A. Hampton and Y. Winter, Eds., *Compositionality and Concepts in Linguistics and Psychology*, ser. Language, Cognition, and Mind. Cham: Springer International Publishing, 2017, vol. 3.

[29] T. E. Zimmermann, "Compositionality problems and how to solve them," in *The Oxford handbook of compositionality*, ser. Oxford handbooks in linguistics, M. Werning, W. Hinzen, and E. Machery, Eds. Oxford: Oxford Univ. Press, 2012.

[30] B. H. Partee, A. T. Meulen, and R. E. Wall, *Mathematical Methods in Linguistics*, 1st ed., ser. Studies in Linguistics and Philosophy Ser. Dordrecht: Springer Netherlands, 1990, vol. v.30.

[31] M. Kracht, "Compositionality: The very idea," *Research on Language and Computation*, vol. 5, no. 3, pp. 287–308, 2007.

[32] M. Fowler, "Dealing with roles."

[33] B. L. Whorf and P. Krausser, Eds., *Sprache, Denken, Wirklichkeit: Beiträge zur Metalinguistik und Sprachphilosophie*, 25th ed., ser. Rowohlts Enzyklopädie. Reinbek bei Hamburg: Rowohlt, 2008, vol. 55403.

[34] L. Boroditsky, "Linguistic relativity," in *Encyclopedia of cognitive science*, L. Nadel, Ed. Chichester, West Sussex Eng. and Hoboken, N.J: Wiley, 2005.

[35] ——, "How language shapes thought," *Scientific American*, vol. 304, no. 2, pp. 62–65, 2011.

[36] R. K. Hendricks and L. Boroditsky, "New space-time metaphors foster new nonlinguistic representations," *Topics in Cognitive Science*, vol. 9, no. 3, pp. 800–818, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/tops.12279

[37] O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, *Structured programming*, 11th ed., ser. APIC studies in data processing. London: Academic Press, 1972, vol. 8.

[38] M. Kracht, "The emergence of syntactic structure," *Linguistics and Philosophy*, vol. 30, no. 1, pp. 47–95, 2007.

[39] ——, *The Mathematics of Language*. UCLA, 2003. [Online]. Available: https://linguistics.ucla.edu/people/Kracht/courses/compling2-2007/formal.pdf

[40] Cantor, "Ueber unendliche, lineare punktmannichfaltigkeiten. 5. fortsetzung: Fortsetzung des artikels in bd. xxi, pag 51." *Mathematische Annalen*, vol. 21, pp. 545–591, 1883. [Online]. Available: https://eudml.org/doc/157080

[41] M. Kracht, "Using each other's words," in *The Road to Universal Logic*. Birkhäuser, Cham, 2015, pp. 341–349. [Online]. Available: https://rd.springer.com/chapter/10.1007/978-3-319-10193-4_15

[42] D. Hilbert and W. Ackermann, *Grundzüge der Theoretischen Logik*, 6th ed., ser. Grundlehren der Mathematischen Wissenschaften Ser. Berlin, Heidelberg: Springer Berlin / Heidelberg, 1972, vol. v.27.

[43] T. Berners-Lee and M. Fischetti, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*, 1st ed. San Francisco, Calif.: HarperCollins, 2000.

[44] R. Koschke, J.-F. Girard, and M. Wurthner, "An intermediate representation for integrating reverse engineering analyses," in *Proceedings / Fifth Working Conference on Reverse Engineering*. Los Alamitos, Calif.: IEEE Computer Society Press, 1998, pp. 241–250.

[45] H. P. Barendregt, M. C. J. D. van Eekelen, J. R. W. Glauert, J. R. Kennaway, M. J. Plasmeijer, and M. R. Sleep, "Term graph rewriting," in *PARLE*, ser. Lecture Notes in Computer Science, J. W. de Bakker, A. J. Nijman, P. C. Treleaven, and J. W. de Bakker, Eds. Berlin: Springer, 1987, pp. 141–158.

[46] D. Plump, "Term graph rewriting," in *Applications, languages and tools*, ser. Handbook of graph grammars and computing by graph transformation / /managing ed, H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg, H. Ehrig, and G. Rozenberg, Eds. Singapore: WORLD SCIENTIFIC, 1999, pp. 3–61.

[47] J. Goll, *Architektur- und Entwurfsmuster der Softwaretechnik: Mit lauffähigen Beispielen in Java*, 2nd ed. Wiesbaden: Springer Vieweg, 2014.

[48] D. Bäumer, D. Riehle, W. Siberski, and M. Wulf, "The role object pattern," in *Washington University Dept. of Computer Science*, 1998.

[49] C. Rupp and S. Queins, *UML2 glasklar: Praxiswissen für die UML-Modellierung*, 4th ed. München: Hanser, 2012. [Online]. Available: http://www.hanser-elibrary.com/action/showBook?doi=10.3139/9783446431973

[50] H. Balzert, *Lehrbuch der Objektmodellierung: Analyse und Entwurf ; mit CD-ROM*, ser. Lehrbücher der Informatik. Heidelberg and Berlin: Spektrum Akad. Verl., 1999.

[51] P. Coad, "Object-oriented patterns," *Communications of the ACM*, vol. 35, no. 9, pp. 152–159, 1992.

[52] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing - STOC '71*, M. A. Harrison, R. B. Banerji, and J. D. Ullman, Eds. New York, New York, USA: ACM Press, 1971, pp. 151–158.

[53] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "A general path-based representation for predicting program properties," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 404–419.

[54] ——, "code2vec: Learning distributed representations of code," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, 2019.

[55] H. Yarahmadi and S. M. H. Hasheminejad, "Design pattern detection approaches: a systematic review of the literature," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5789–5846, 2020.

[56] R. Compton, E. Frank, P. Patros, and A. Koay, "Embedding java classes with code2vec: Improvements from variable obfuscation," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 243–253.

[57] G. Rasool, I. Philippow, and P. Mäder, "Design pattern recovery based on annotations," *Advances in Engineering Software*, vol. 41, no. 4, pp. 519–526, 2010.

[58] P. Warintarawej, M. Huchard, M. Lafourcade, A. Laurent, and P. Pompidor, "Software understanding: Automatic classification of software identifiers," *Intelligent Data Analysis*, vol. 19, no. 4, pp. 761–778, 2015.

[59] M. Schindler, A. Rausch, and O. Fox, "Clustering source code elements by semantic similarity using wikipedia," in *Proceedings of 4th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, 2015, pp. 13–18.

[60] J. He, C.-C. Lee, V. Raychev, and M. Vechev, "Learning to find naming issues with big code and small supervision," in *2021 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI '21)*. ACM, 2021, pp. 1–16.

[61] M. Schindler and A. Rausch, "Architectural concepts and their evolution made explicit by examples," in *ADAPTIVE 2019, The Eleventh International Conference on Adaptive and Self-Adaptive Systems and Applications*, vol. 11, 2019, pp. 38–43.

[62] jqassistant — your software . your structures . your rules. [retrieved: 03, 2023]. [Online]. Available: https://jqassistant.org

[63] Unit test your java architecture - archunit. [retrieved: 03, 2023]. [Online]. Available: https://www.archunit.org

[64] Structure101 software architecture development environment (ade). [retrieved: 03, 2023]. [Online]. Available: https://structure101.com

[65] Dependometer. [retrieved: 03, 2023]. [Online]. Available: https://github.com/dheraclio/dependometer

[66] G. Rasool and D. Streitfdert, "A survey on design pattern recovery techniques," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 6, p. 251, 2011.

[67] W. Wang, G. Li, B. Ma, X. Xia, and Z. Jin, "Detecting code clones with graph neural network and flow-augmented abstract syntax tree," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 261–271.

[68] C. Knieke, A. Rausch, and M. Schindler, "Tackling software architecture erosion: Joint architecture and implementation repairing by a knowledge-based approach," in *2021 IEEE/ACM International Workshop on Automated Program Repair (APR)*. IEEE, 6/1/2021 - 6/1/2021, pp. 19–20.

[69] C. Knieke, K. Marco, A. Rausch, M. Schindler, A. Strasser, and M. Vogel, "A holistic approach for managed evolution of automotive software product line architectures," in *ADAPTIVE 2017*, A. A. Enescu and A. Rausch, Eds. Wilmington, DE, USA: IARIA, 2017, pp. 43–52.

[70] B. Bollobás, *Modern Graph Theory*. New York, NY: Springer New York, 1998, vol. 184.

[71] H. Lee-Kwang and K.-M. Lee, "Fuzzy hypergraph and fuzzy partition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 1, pp. 196–201, 1995.

[72] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*, 3rd ed., ser. Safari Tech Books Online. Upper Saddle River, NJ: Addison-Wesley, 2013.

[73] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects*, 1st ed., ser. Wiley Software Patterns Series. s.l.: Wiley, 2013. [Online]. Available: http://gbv.eblib.com/patron/FullRecord.aspx?p=699910

[74] S. Herold, *Architectural compliance in component-based systems: Foundations, specification, and checking of architectural rules*, 1st ed., ser. SSE-Dissertation. München: Verl. Dr. Hut, 2011, vol. 5.

[75] S. Herold, C. Knieke, M. Schindler, and A. Rausch, "Towards improving software architecture degradation mitigation by machine learning," in *ADAPTIVE 2020, The Twelfth International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2020, pp. 36–39.

VR-SysML+Traceability: Immersive Requirements Traceability and Test Traceability with SysML to Support Verification and Validation in Virtual Reality

Roy Oberhauser[0000-0002-7606-8226]
Computer Science Dept.
Aalen University
Aalen, Germany
e-mail: roy.oberhauser@hs-aalen.de

Abstract - **As systems grow in complexity, the interdisciplinary nature of systems engineering makes the visualization and comprehension of the underlying system models challenging for the various stakeholders. This, in turn, can affect validation and realization correctness. Furthermore, stakeholder collaboration is often hindered due to the lack of a common medium to access and convey these models, which are often partitioned across multiple 2D diagrams. This paper contributes VR-SysML, a solution concept for visualizing and interacting with Systems Modeling Language (SysML) models in Virtual Reality (VR). Our prototype realization shows its feasibility, and our evaluation results based on a case study shows its support for the various SysML diagram types in VR, cross-diagram element recognition via our Backplane Followers concept, and depicting further related (SysML and non-SysML) models side-by-side in VR.**

Keywords - Systems Modeling Language (SysML); virtual reality; systems modeling; systems engineering; requirements traceability; test traceability; system testing; verification and validation.

I. INTRODUCTION

This paper extends the immersive Systems Modeling Language (SysML) model visualization and interaction capabilities in VR-SysML [1]. Towards supporting immersive software (SW) verification and validation (V&V), it contributes semi-automated requirements traceability and test tracing capabilities visualized in VR.

Systems engineering (SysE) is an interdisciplinary collaborative engineering field dealing with the design, integration, and management of complex system solutions over their lifecycle. The field faces a continuous challenge of growing system complexity, an increasing share of functionality shifted to software, system resource constraints, while coping with compressed development timeframes and project budget and resource constraints. Furthermore, the interdisciplinary nature of SysE means that diverse stakeholder types and groups with their specialty competencies and concerns are involved and who may not be readily acquainted with the model types and modeling languages involved. Any models may be digitally isolated or practically inaccessible to all stakeholder types, "hidden" within "cryptic" modeling tools that certain modeling specialists may understand. Due to the interdisciplinary nature of SysE, the inaccessibility and lack of model comprehension can hamper collaboration and affect overall system validity and correctness with regard to requirements. Visualized requirements traceability can help support validity checking. Furthermore, visualizing test traceability can help with the analysis of the testing effort and support V&V.

While SysE can involve various models including physical, mechanical, electrical, thermodynamic, and electronic, the focus of this paper is on the Systems Modeling Language [2]. SysML is a dialect of the Unified Modeling Language (UML®) and defined as a UML 2 Profile. Views and their associated diagrams can help reduce cognitive overload, yet their divided nature also risks overlooking a relation or element and comprehending the overall model. Ideally, a model should be whole and complete to the appropriate degree for the reality it is depicting and simplifying. Yet the modeling languages and associated tooling typically assumes a 2D display and portrays portions of models sliced onto 2D diagrams. Although 3D models can be portrayed on 2D displays, they lack an immersion quality.

VR is a mediated visual environment which is created and then experienced as telepresence by the perceiver. VR provides an unlimited immersive space for visualizing and analyzing a growing and complex set of system models and their interrelationships simultaneously in a 3D spatial structure viewable from different perspectives. Lacking a proper 3D system modeling notation, in the interim we propose retaining the well-known SysML notation and interconnecting 2D SysML diagrams in VR, which can suffice for depicting the relations between elements across diagrams and assist with navigating and validating complex models. As system models grow in complexity and reflect the deeper integration and portrayal of their system reality and environment, an immersive digital environment provides an additional visualization capability to comprehend the "big picture" model for structurally and hierarchically complex system models via interconnected diagrams and associated digital elements.

As to our prior work in visualizing architecture in VR, VR-UML [3] provides VR-based visualization and interaction with UML models. VR-EA [4] visualizes Enterprise Architecture (EA) ArchiMate models in VR. Extending VR-SysML [1], this paper contributes VR-SysML+Traceability, a VR-based solution concept for visualizing and interacting with SysML while adding additional SysML automated requirements traceability and test tracing capabilities to support SW V&V. Our prototype realization shows its

feasibility, and a case-based evaluation provides insights into its capabilities.

The remainder of this paper is structured as follows: Section 2 discusses related work. In Section 3, the solution concept is described. Section 4 provides details about the realization. The evaluation is described in Section 5 and is followed by a conclusion.

II. Related Work

As to visualization approaches with SysML, Nigischer and Gerhard [5] proposed a lightweight 3D visualization for SysML models in Product Data Management. They describe an approach and concept, but no prototype is shown. Barosan et al. [6] describes a 3D SysML digital-twin-in-loop virtual simulation environment of a distribution center for truck driving test scenarios integrating IBM Rhapsody with Unity3D; VR and immersion are not considered. Mahboob et al. [7] describe a model-based approach to generate VR object collision simulation scenes from SysML behavior models.

Besides our own VR-UML [3], VR features are not yet commonplace in UML tools: Ozkaya [8] analyzed 58 different UML tools without any mention of VR, and Ozkaya and Erata [9] surveyed 109 practitioners to determine their UML preferences without any mention of VR. Non-VR 3D-based UML visualization includes X3D-UML [10], VisAr3D [11], and the case study by Krolovitsch and Nilsson [12].

Work related to requirements traceability visualization includes Li & Maalej [13], which found traceability matrices and graphs preferrable for management tasks. Matrices were preferred for an overview, while graphs were preferred for navigating linked artifacts. They noted that users were not always capable of choosing the most suitable traceability visualization. Abad et al. [14] performed a systematic literature review on requirements engineering visualization. Madaki & Zainon [15] performed a review on tools and techniques for visualizing SW requirement traceability. None of the above literature mentioned immersive or VR techniques; our literature search did not find similar work.

With regard to test traceability, we found no VR work directly addressing this topic. VR-related work regarding software analysis includes VR City [16], which applies a 3D city metaphor. While it briefly mentions that its work might be used for test, it shows no actual results in this regard and in this regard only a trace mode visualization is depicted.

In contrast, VR-SysML+Traceability provides an immersive visualization and experience with SysML models, providing automatic layout of views as stacked 3D hyperplanes, visualizing the reality of inter-view relations and recurrence of elements, and enabling interactive modeling in VR. Furthermore, it provides traceability of requirements and test status to immersively support V&V. Hypermodeling support enables SysML, UML, and other relevant models to be simultaneously visualized in the same virtual space, supporting cross-model analysis across various diagram types and stakeholder concerns.

III. Solution Concept

Our solution concept is based on VR. In support of our view that an immersive VR experience can be beneficial for model analysis, Müller et al. [17] compared VR vs. 2D for a software analysis task, finding that VR does not significantly decrease comprehension and analysis time nor significantly improve correctness (although fewer errors were made). While interaction time was somewhat less efficient than the common daily 2D interactions one is used to and has been trained in for years, it is important to note that VR improved the user experience, was more motivating, less demanding, more inventive/innovative, and more clearly structured.

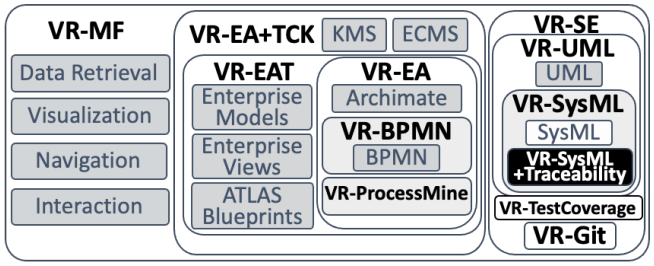


Figure 1. Conceptual map of our various VR solution concepts.

SysML is a general-purpose architecture modeling language for systems and systems-of-systems, supporting their specification, analysis, design, verification, and validation. Out of UML 2's diagrams, it reuses seven (modifying four of these) while adding two additional ones. Thus, for VR-SysML (Figure 1) we chose to extend our VR-UML [3] solution concept, which is based on our generalized VR Modeling Framework (VR-MF) (detailed in [4]). VR-MF provides a VR-based domain-independent hypermodeling framework addressing four aspects requiring special attention when modeling in VR: visualization, navigation, interaction, and data retrieval. Our other VR architectural modeling solutions include VR-BPMN [18], VR-ProcessMine [19], VR-EA [4], and VR-EAT [20], which integrates the EA tool Atlas to provide dynamically-generated EA diagrams in VR. VR-EA+TCK [21] adds additional capabilities, integrating enterprise Tool, Content, and Knowledge such as a Knowledge Management Systems (KMS) and/or Enterprise Content Management Systems (ECMS). While SysML is popular for embedded and model-based systems, it is also applicable to domains such as EA. In the Software Engineering (SE) area, which we group under VR-SE, whereby VR-TestCoverage [22] and VR-Git [23] address test coverage and code repository aspects in VR.

A. Visualization in VR

Our concept attempts to leverage the best of 2D and VR: to support diagram comprehension, we chose not to diverge significantly from the SysML notation. Yet placing 2D SysML images like flat screens in front of users would provide little added value in the 3D VR space. A plane is used to intuitively represent a diagram. Stacked hyperplanes are used to support viewing multiple diagrams at once, while permitting a user to readily have an overview of the number and types of diagrams. Furthermore, hyperplanes serve a grouping function and allow us to utilize the concept of a common transparent or invisible backplane to indicate common elements across diagrams via multi-colored inter-

diagram followers. Versus side-by-side, stacked diagrams are a scalable approach for larger projects since the distance to the VR camera is shorter. Multiple stacks can be used to group diagrams or delineate heterogeneous models. Diagrams of interest can still be viewed side-by-side by moving them from the stack via an anchor sphere affordance on a diagram corner, which is also used to hide or collapse diagrams to reduce visual clutter. To distinguish SysML elements types, 2D icon images can be placed on generic (e.g., block) model elements, in order to reduce the effort of modeling each SysML element type as a separate 3D form for VR.

B. Navigation in VR

One navigation challenge arising from the immersion VR offers is supporting intuitive spatial navigation while reducing potential VR sickness symptoms. Thus, we incorporate two navigation modes in our solution concept: the default uses gliding controls for fly-through VR, while teleporting instantly places the camera at a selected position. Although potentially disconcerting, it may reduce the likelihood of VR sickness induced by fly-through for those prone to it.

C. Interaction in VR

As VR interaction has not yet become standardized, in our concept user-element interaction is supported primarily through VR controllers and a *Virtual Tablet*. The VR-Tablet provides detailed element information with context-specific Create, Retrieve, Update, Delete (CRUD) capabilities including a *virtual keyboard* for text entry via laser pointer key selection. The aforementioned corner *anchor sphere* affordance supports moving / hiding / displaying diagrams. Inter-diagram element *followers* can be displayed, hidden, or selected (emphasized).

D. Traceability

A modeling tool such as Sparx Systems Enterprise Architect can be used to provide requirement and test traceability information via SysML. Our solution then extracts traceability-related information from relevant SysML diagrams (Requirements and/or Use Case diagrams), including elements such as Requirement (stereotype «requirement») and Test Case, and relations such as «satisfy», «verify», and «deriveReqt», etc.

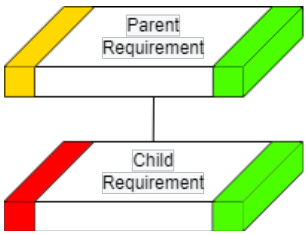


Figure 2. Depicting (sub/super)-requirement dependencies, with degree of implementation on the left and test implementation on the right edge.

For tracing requirements with their dependent code implementation status, annotations are placed in the code to indicate with requirement IDs are addressed. These are extracted by a tool that parses all (test) code files and generates a report. The result is then visualized on the relevant requirement element edges in the diagram in VR (e.g., red means that requirement ID was not found in the (test) code, green if at least one reference was found, and for parent requirements yellow if partially addressed based on some child element(s) missing a reference (see Figure 2). Also, a total degree of implementation considering the elements on that diagram level is also provided on the side of a diagram.

To trace test results to their requirements, the test results are extracted from a test tool report, e.g., in the JUnit XML format used by pytest and JUnit. The test result is then visualized on the relevant requirement diagram elements in VR (e.g., red if no test was found, yellow if not all passed, and green if passed). Also, a total degree of test implementation considering the elements on that diagram level is also provided on the diagram side.

Connectors can be followed to trace these to the actual artifacts, the content of which can be shown in the VR-Tablet.

Note that while the traceability model utilizes information from SysML in addition to other sources, we chose to visualize it in VR independent of SysML conformant constraints, opting for a more intuitive visual depiction of traceability for the stakeholder. Placing a VR-SysML model next to a traceability model is intended and supported. That way, the VR-SysML expresses the exact model it does in a SysML tool, while the traceability model can include the additional automatically extracted implementation and test features without encumbering the SysML model.

For our traceability visualization, we thus chose to layer the information ordered by degree of abstraction as shown in Figure 3. The top layers are SysML model-related: Use cases being the highest abstraction and thus on top, requirements being more concrete and a level below, and test cases being used to verify requirements and thus below requirements but shifted to the side to indicate they relate to testing. The lower layers consist of file trees that visualize the test source code files implementing test cases, and the implementation layer consisting of source code files that implement the requirements.

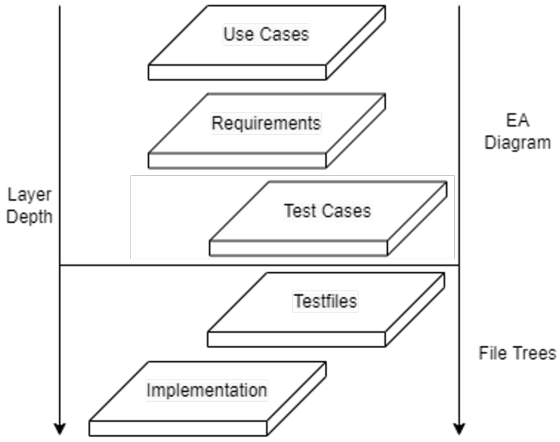


Figure 3. VR-SysML visual layering for traceability.

In general, a backplane is used with colored trace lines to show all the traces, thus indicating the total available traces and the degree of traceability. This avoids a spider web-like

tracing of lines across all elements. However, when an element of interest is selected, then direct trace lines specific to that element are also depicted (analogous to a spotlight).

IV. Realization

The realization of the solution focused on two aspects: 1) realizing a correct portrayal of the various SysML diagrams and providing a way to trace the same element to its occurrence in other diagrams (which we name VR-SysML), and 2) the extraction and visualization of requirements traceability information from SysML diagrams and implementation files and test code files.

A. VR-SysML Realization

The logical architecture for our VR-SysML prototype realization is shown in Figure 4.

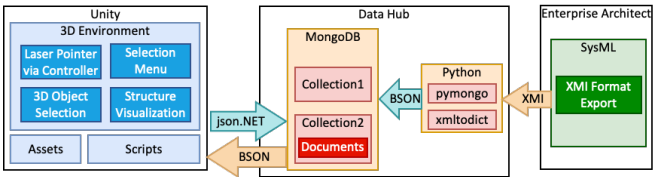


Figure 4. VR-SysML logical architecture.

SysML models are imported in XMI format to our Data Hub that is implemented in Python. Xmitodict is used to convert the XMI to a key-value dictionary and the built-in JSON package is used for JSON conversion. Pymongo is used to store the JSON (as BSON) in the NoSQL document database MongoDB. The scripts in the Unity environment utilize json.NET. SysML XMI files produced from SparxSystems Enterprise Architect were used. Our prototype currently does not consider the Allocation Table (relationship matrices).

B. Requirements Traceability Realization

The logical architecture for our traceability realization is shown in Figure 5.

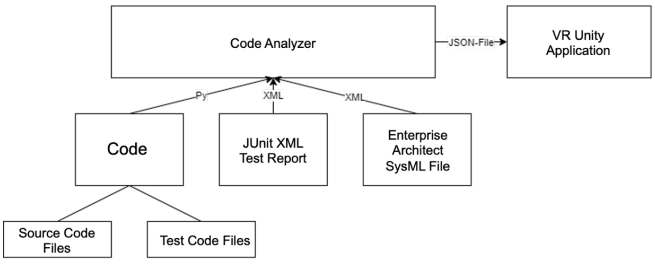


Figure 5. VR-SysML logical architecture for traceability realization.

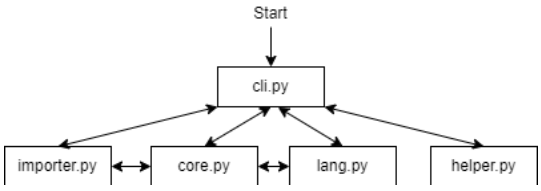


Figure 6. Python modules contained in the Code Analyzer tool.

The Code Analyzer tool has a Command Line Interface (CLI) and is implemented in Python. Based on input parameters, it scans the given files and extracts information such as requirements IDs from code, test reports in JUnit XML, and SysML Enterprise Architect XMI files, producing a JSON file as output that is then imported by the VR application running on the Unity platform. Its modular realization is shown in Figure 6.

Within SysML models, a diagram element with the property "id" serves as the reference for identifying and differentiating requirements as shown in Figure 7 and for test cases as shown Figure 8. The Python library xml.dom.minidom is used to extract the ID, element types, element position and size, relations between elements, and properties, comments, or annotations.

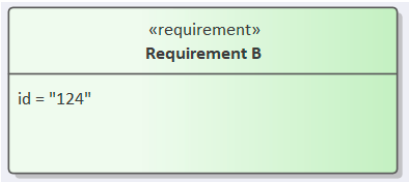


Figure 7. Requirement "id" as property in SysML.

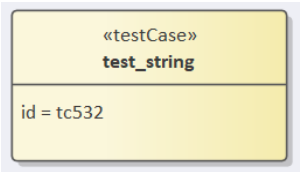


Figure 8. Test case "id" as property in SysML.

```
1 def test_method(): # REQID: 123, 312 TESTID: tc1232 TESTTARGET:
      ↪ network.py
2     pass
```

Figure 9. Traceability annotation example: associating a test method in test code to a requirement and test target (the implementation).

```
1 """Test File"""
2 # REQID: 123
3
4 def test_method():
5     pass
```

Figure 10. Traceability annotation example: associating all methods in a test source file to one (or more) requirement(s).

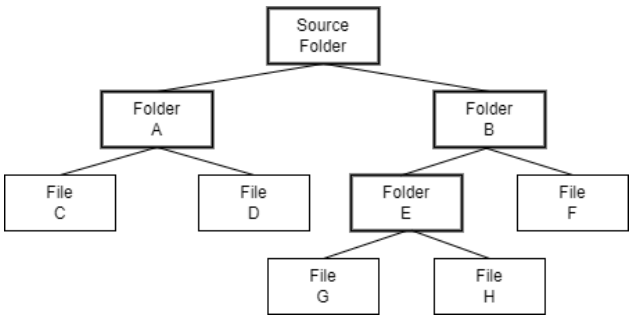


Figure 11. Example file tree with folders.

Since almost all programming languages support comments, the annotations are provided as comments and can thus be utilized in any programming language. For this, lang.py must be extended for each additional language. Within source code, the keyword “REQID” indicates the unique identifier (ID) of a requirement, and can be associated with a (test) method, as shown in Figure 9. When a (test) method satisfies multiple requirements, multiple IDs can be separated via commas. The keyword "TESTID" is used to associate a SysML diagram. Finally, “TESTTARGET” is only used within test files to indicate the test target. Multiple references are possible separated by commas. Some requirements are overarching and it would be arduous to associate each method separately. Thus, for the case when all methods in a file address one (or more) requirement(s), “REQID” can be placed at the top of the file (separate from method declarations or definitions), thus implicitly indicating it is associated with all methods in that file, as exemplified in Figure 10.

To build balanced file trees to portray the test source and implementation source, with each tree consisting of folders and files (see Figure 11), an implementation of Reingold-Tilford algorithm [24] was adapted.

V. Evaluation

We base the evaluation of our solution concept on design science method and principles [25].

A. VR-SysML Evaluation

A case study is used with an emphasis on SysML diagram type support, how these are visualized in VR, and additional capabilities in VR. A sample SysML project with all 9 SysML diagram types is used to compare the visualization in Enterprise Architect to that in VR-SysML, grouped as requirement, behavior, or structure diagram types.

As shown in Figure 12, the various diagrams of the SysML model are mapped to stacked hyperplanes that provide an anchor affordance (black sphere) with which to expand, collapse, or move a diagram. Planes and elements have a shallow 3D depth with labeled edges to support recognition from different viewing angles. The colors of the planes can be configured to help with differentiation or grouping. Furthermore, our backplane concept creates followers that allow one to quickly find the same element across different diagrams in the same model, to readily see in which diagrams that element participates, or to determine that the element is only shown on one diagram (it not having a follower). The colored followers can be selected (made bold) and the other followers can be hidden if desired to reduce visual clutter for larger models.

1) SysML Requirement Diagram. SysML extends UML with an additional diagram type, the Requirement diagram. It can be used to specify functional and non-functional requirements for the model. An example viewed in EA is shown Figure 13 and in VR in Figure 14. In VR, elements are labeled on edges to support reading from different angles. The VR Tablet can provide more details or interaction capabilities for a selected element, and while support for modeling capabilities is shown on the interface, these are currently placeholders and have not yet been fully implemented in the prototype (create, modify, delete, export).

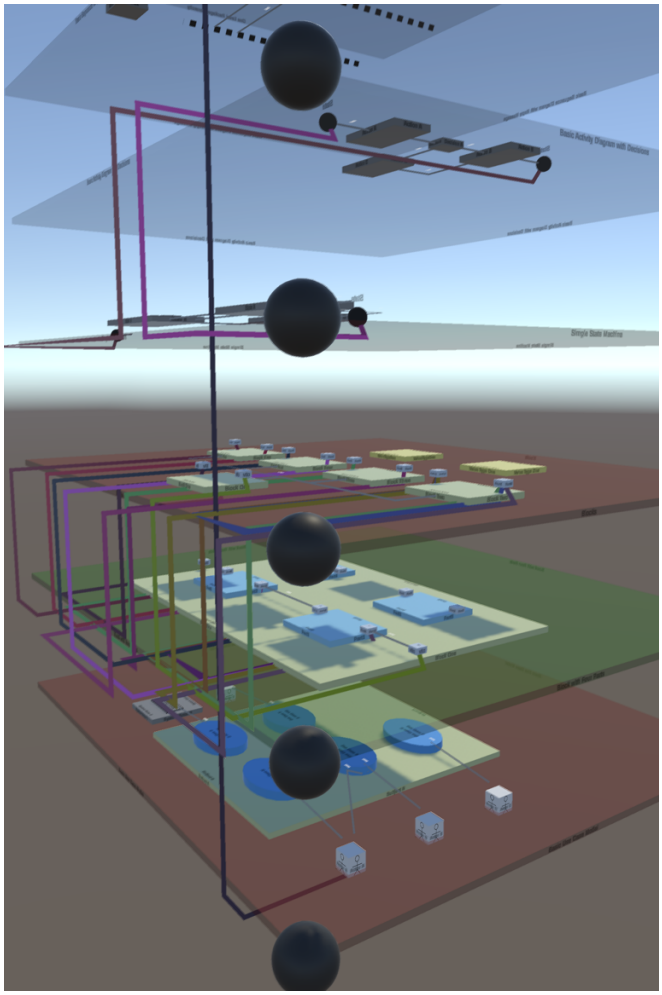


Figure 12. VR-SysML backplane with inter-diagram followers.

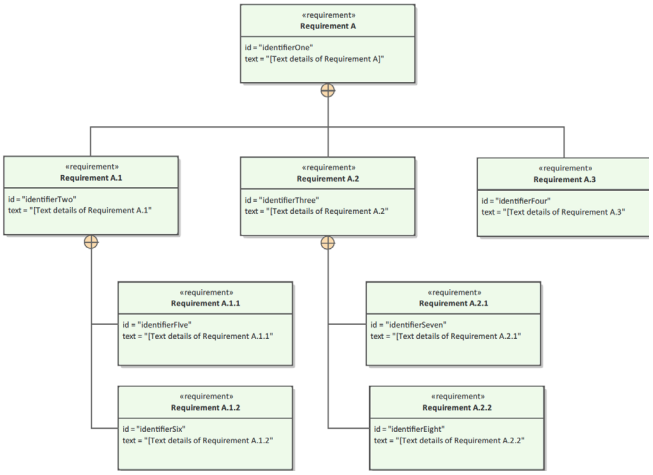


Figure 13. Requirement Diagram in EA.

Figure 14. Requirement Diagram in VR.

2) SysML Use Case Diagram. As a behavior diagram, SysML includes the Use Case Diagram from UML as shown from EA in Figure 15 and VR in Figure 16. In order to more readily recognize and differentiate the diagram type, an oval shape was used for the use cases. However, the actors utilize our generic cube concept with notation symbols placed on the various sides. This provides a flexible mechanism for quickly supporting various notation element types and tailoring or extending model element types using any icons or images.

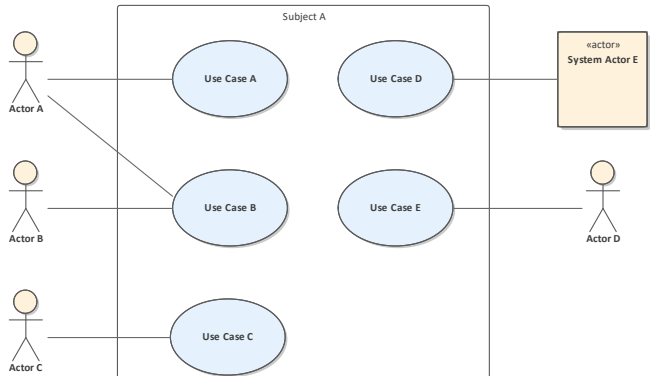


Figure 15. Use Case Diagram in EA.

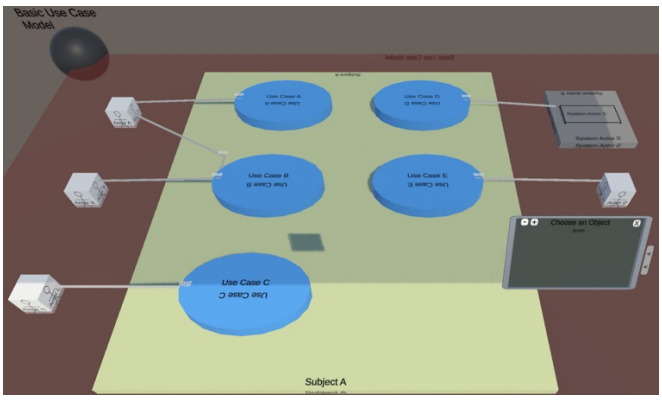


Figure 16. Use Case Diagram in VR.

3) SysML Activity Diagram. Another dynamic behavior diagram type that can be used to specify dynamic system behaviors, such as control flow and object flows, is the Activity diagram in SysML from EA in Figure 17 and VR in Figure 18. It is slightly modified from that in UML, adding additional semantics for Continuous Flow and Probability.

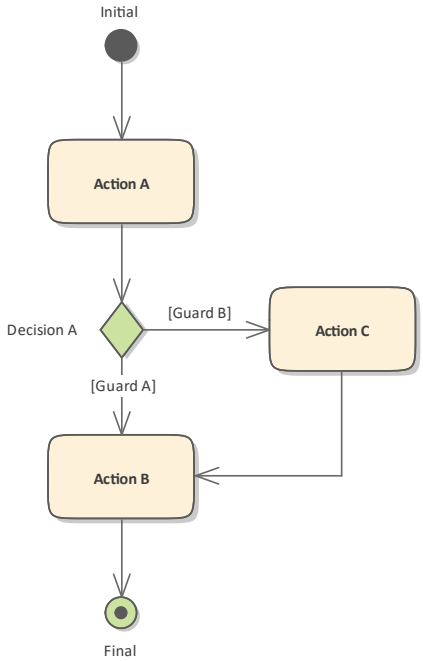


Figure 17. Activity Diagram in EA.

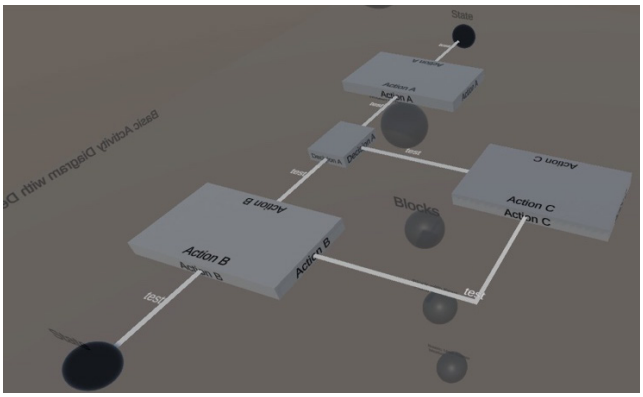


Figure 18. Activity Diagram in VR.

4) SysML Sequence Diagram. Sequence diagrams (unmodified from UML) provide a further dynamic behavior diagram, showing interactions via message sequences, from EA in Figure 19 and VR in Figure 20.

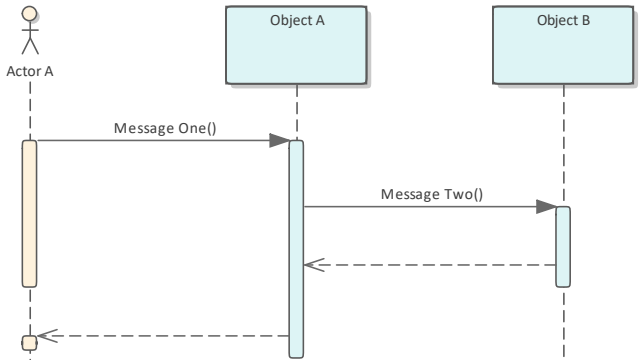


Figure 19. Sequence Diagram in EA.

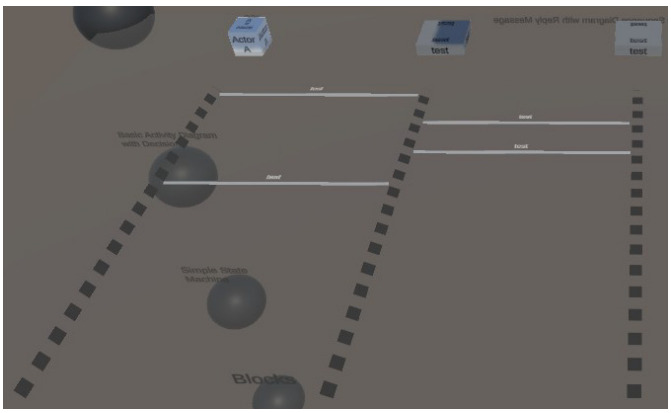


Figure 20. Sequence Diagram in VR.

5) SysML State Machine Diagram. State machine diagrams (unmodified from UML) are a dynamic behavior diagram showing states transitions that occur in response to events, from EA in Figure 21 and VR in Figure 22.

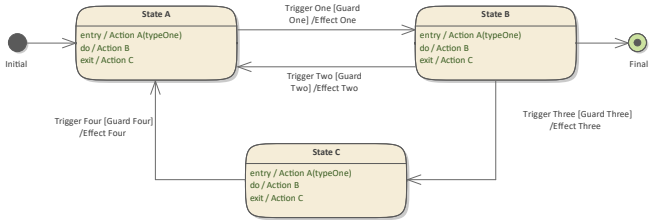


Figure 21. State Machine Diagram in EA.

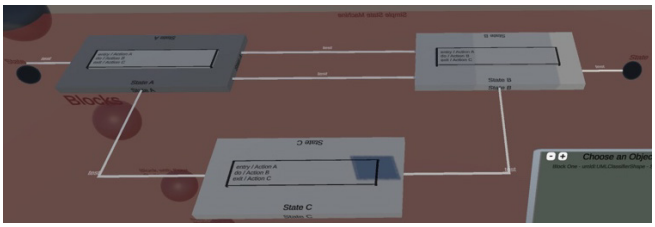


Figure 22. State Machine Diagram in VR.

6) SysML Block Definition Diagram (BDD). A BDD is a static structural diagram, analogous to the UML Class diagram type with certain modifications, and shows system components, their contents (as properties, behaviors, constraints), interfaces, and relationships. See Figure 23 for an example from EA and Figure 24 for VR. It can be used for describing the system structure as a hierarchy of relations between systems and subsystems typically consisting of "black-box" blocks. As a possible specialization, it can be useful to explicitly model constraints separately, referred to as Constraint Block diagrams (see Figure 25 for an EA example and Figure 26 for VR), which can be referenced by Parametric diagrams.

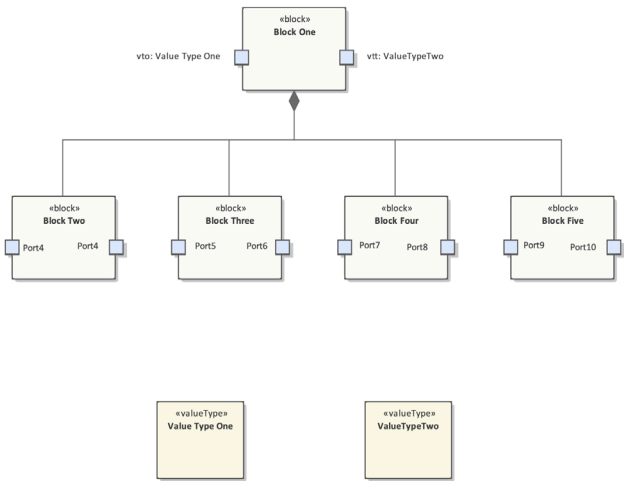


Figure 23. Block Definition Diagram (BDD) in EA.

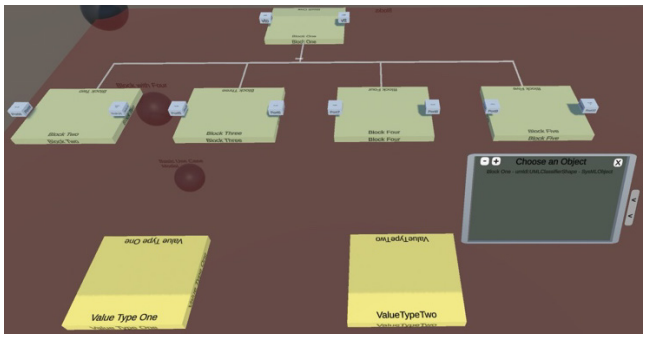


Figure 24. Block Definition Diagram (BDD) in VR

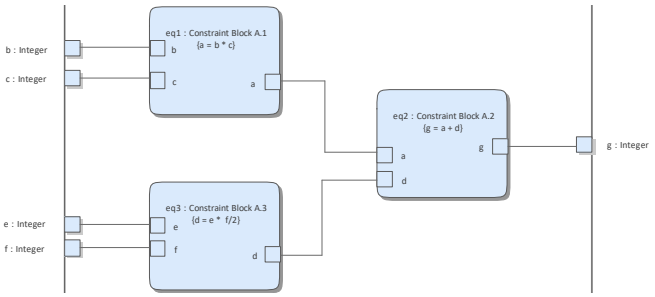


Figure 25. Constraint Block Diagram in EA.

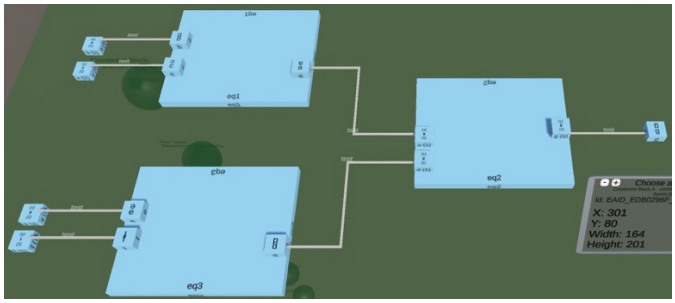


Figure 26. Constraint Block Diagram in VR.

7) SysML Internal Block Diagram (IBD). An IBD is a static structural diagram that depicts the internal (encapsulated) composition (structural contents) of a Block in a BDD, i.e., a "white-box" view. This includes properties, parts, interfaces, connectors, and ports, and can be used to depict the flow of inputs and outputs between them. See Figure 27 for an example in EA and Figure 28 for VR.

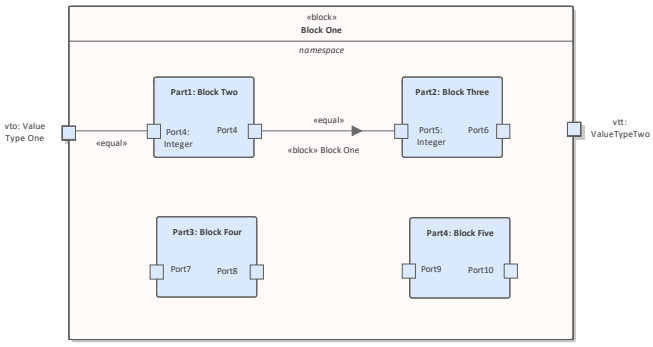


Figure 27. IBD in EA.

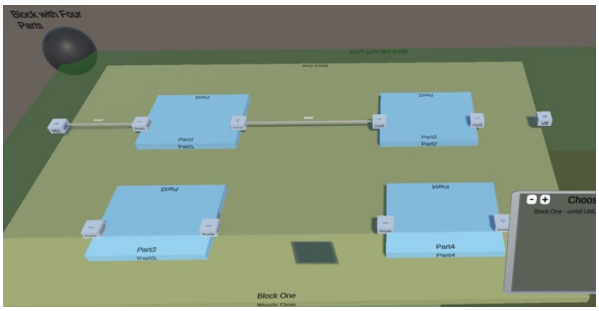


Figure 28. IBD in VR.

8) SysML Parametric Diagram. A static structural diagram type, Parametric diagrams (see Figure 29 for EA and Figure 30 for VR) are a specialization of IBD to model equations with parameters and can be used to enforce mathematical rules or constraints defined via Constraint Blocks.

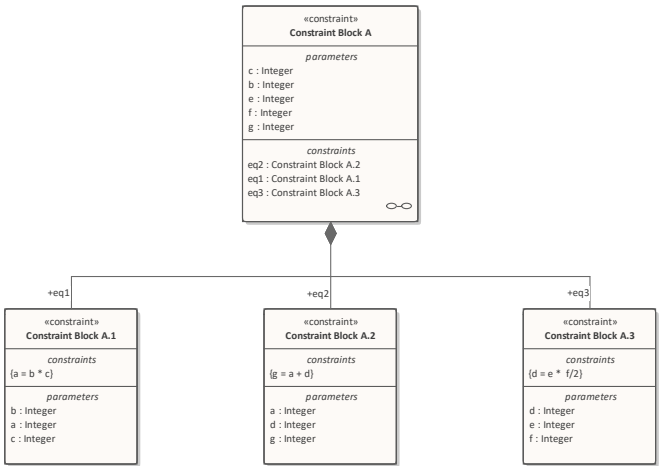


Figure 29. Constraint Parametric Diagram in EA.

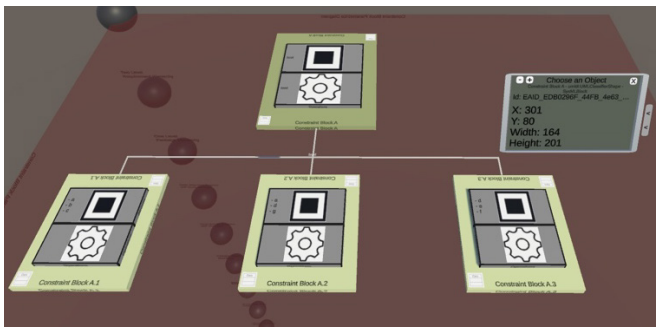


Figure 30. Constraint Parametric Diagram in VR.

9) SysML Package Diagram. A SysML Package diagram (see Figure 31 for EA and Figure 32 for VR) is further static structural diagram based on the equivalent UML type (with minor modifications). Packages provide a general-purpose mechanism for grouping model elements and diagrams, and the diagram can be used to show their contents and the relationship between them.

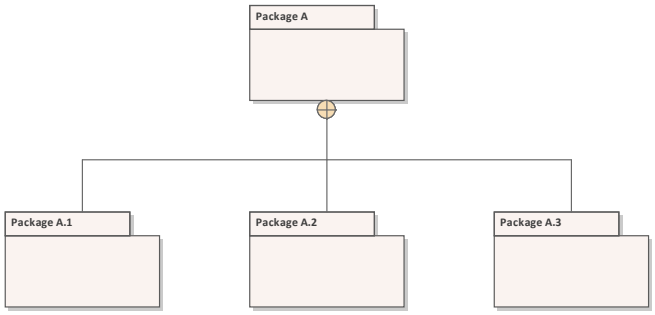


Figure 31. Package Diagram in EA.

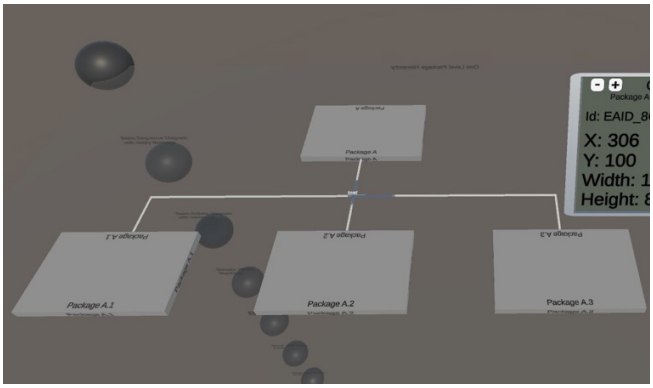


Figure 32. Package Diagram in VR.

10) Multi- and Heterogeneous Model Depiction in VR. VR's unlimited virtual space provides the potential to view, compare, and analyze multiple SysML (left and center models in Figure 33) or heterogeneous models side-by-side, exemplified with an ArchiMate enterprise architecture model on the right in Figure 33. For SysE, this immersive approach also has the potential to support interdisciplinary collaboration between specialization experts for complex systems.

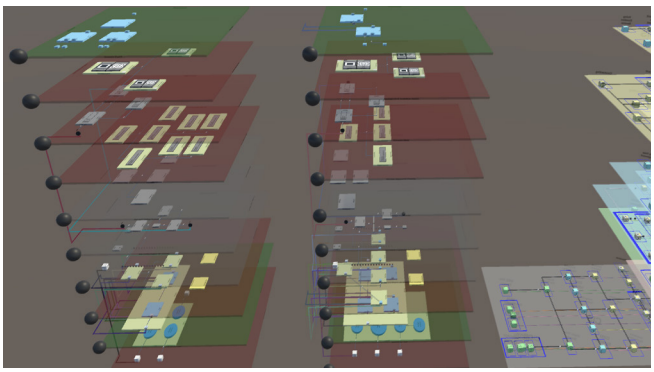


Figure 33. Multiple and heterogeneous side-by-side models in VR.

B. *Traceability Scenario: Requirements and Tests*

To evaluate the traceability scenario, an example project is used, consisting of a various source and test files with a SysML requirements diagram in EA (see Figure 34). As can be seen in the diagram, use cases related to requirements via satisfy, and further child requirements via derived relations, and test cases having a verify relation to the requirements.

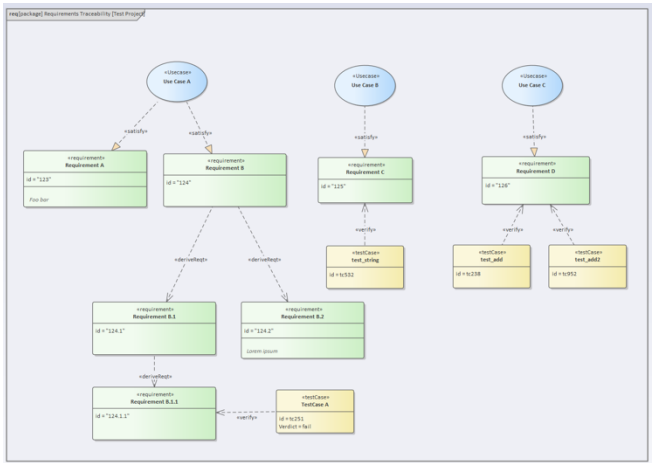


Figure 34. Example requirements diagram in EA.

Output of the code analysis tool is shown in Figure 35. A separate JSON file (not shown) with this information is generated for VR to process the information for visualization. The output shows the scanned use cases, derived requirements, test cases, test files, and implementation source files, and their relations.

The visualization of the traceability layers (Figure 36) and relations backplane (Figure 37) are shown in VR. Use cases (biege) are the most abstract at the top (Figure 38), with satisfy relations following requirements. The number of requirements layers (dark grey) (Figure 39) are variable, depending on the depth of derivation hierarchy, in this case two additional layers. The test cases layer (Figure 40) is shown in (blue) and offset to the right of the stack. File trees for the implementation and the tests (shifted to the right) are depicted on the lowest purple planes as seen in Figure 41.

```
25-Jan-22 12:51:52 - INFO: Start requirelyzer.
25-Jan-22 12:51:52 - INFO: Importing Projects...
25-Jan-22 12:51:52 - INFO: Start parsing project: 'Enterprise Architect' (Parser: python)
25-Jan-22 12:51:52 - WARNING: Testfile 'test_example1.py' (538eb418-ce6e-42a4-b551-4812ddbb4de3) already added to 'Requirement D' (126)
25-Jan-22 12:51:52 - WARNING: Testfile 'test_example1.py' (538eb418-ce6e-42a4-b551-4812ddbb4de3) already added to 'Requirement C' (125)
25-Jan-22 12:51:53 - INFO: Import JUnit test file and compare it with the analyzed test cases.
25-Jan-22 12:51:53 - INFO: Parsing done in 0.03743909999999995s
---------- USE CASES ----------
Name
----------
Use Case C
Use Case A
Use Case B
---------- REQUIREMENTS ----------
ID      Name              Priority    Dependencies    Annotation    Testfiles
------  ----------------  ----------  --------------  ------------  ------------------------------
126     Requirement D     Low                                       test_example1.py
123     Requirement A     High                        Foo bar
125     Requirement C     Medium                                    test_example1.py, test_foobar.py
124     Requirement B     Medium      124.2, 124.1                  test_bar.py
124.2   Requirement B.2   High                        Lorem ipsum   test_fail.py
124.1   Requirement B.1   Medium      124.1.1                       test_example1.py
124.1.1 Requirement B.1.1 Medium                                    test_bar.py
---------- DIAGRAM TESTCASES ----------
ID    Name         Requirements    Has Testfile
----  -----------  --------------  --------------
tc251 TestCase A   124.1.1         False
tc532 test_string  125             True
tc238 test_add     126             False
tc952 test_add2    126             True
---------- TESTCASES ----------
ID    Name         Failed    Requirements    Testtargets
----  -----------  --------  --------------  ------------
tc952 test_add     False     124.1, 126, 125  network.py
      test_add2    False     126, 125
      test_failed  True      124.2
      test_string  False     124, 124.1.1
tc532 test_string  False     125              core.py
      test_math    False
---------- TEST FILES ----------
Name              Path                                                                                                  Testcases  Failed    Testcases
----------------  ----------------------------------------------------------------------------------------------------  ---------  --------  --------------------
test_example1.py  D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\tests\test_example1.py                    2  False     test_add, test_add2
test_fail.py      D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\tests\test_fail.py                      1  True      test_failed
test_bar.py       D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\tests\bar\test_bar.py                   1  False     test_string
test_foobar.py    D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\tests\bar\test_foobar.py                1  False     test_string
test_foo.py       D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\tests\foo\test_foo.py                   1  False     test_math
---------- IMPLEMENTATIONS ----------
Name        Path                                                                                  Requirements
----------  ------------------------------------------------------------------------------------  ------------
core.py     D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\src\core.py        124
network.py  D:\Projekte\SE-Projekt\21w-pa-greiner\test-projects\python-project\src\network.py     125,124
25-Jan-22 12:51:53 - INFO: Exporting project file: Enterprise Architect.json
(venv) PS D:\Projekte\SE-Projekt\21w-pa-greiner\code-analyzer>
```

Figure 35. Code analyzer output.

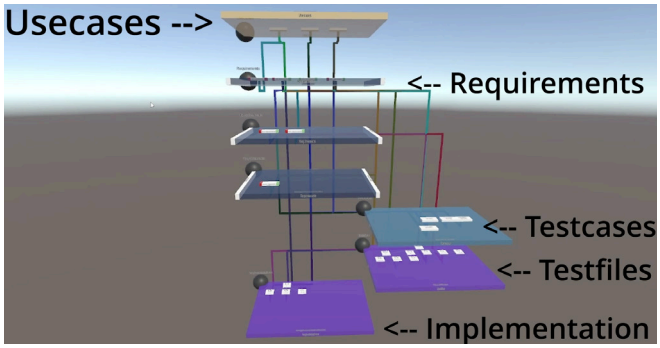


Figure 36. Annotated traceability layers and relations backplane in VR (without element selected).

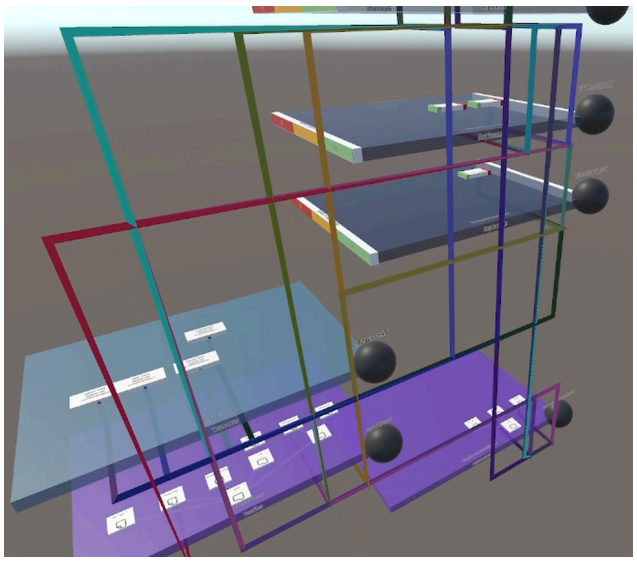


Figure 37. Traceability relations on backplane.

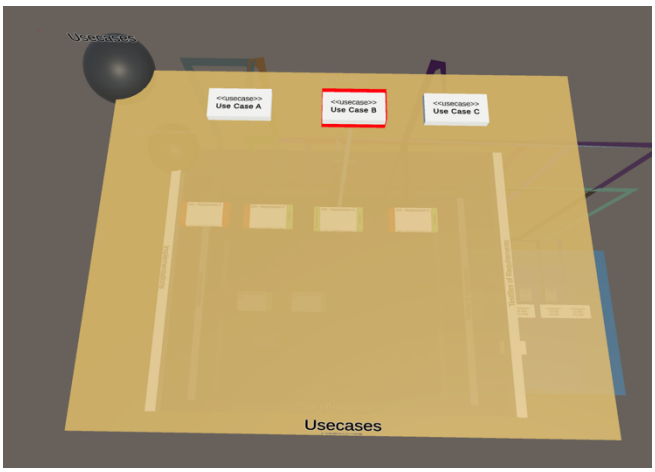


Figure 38. Use case layer.

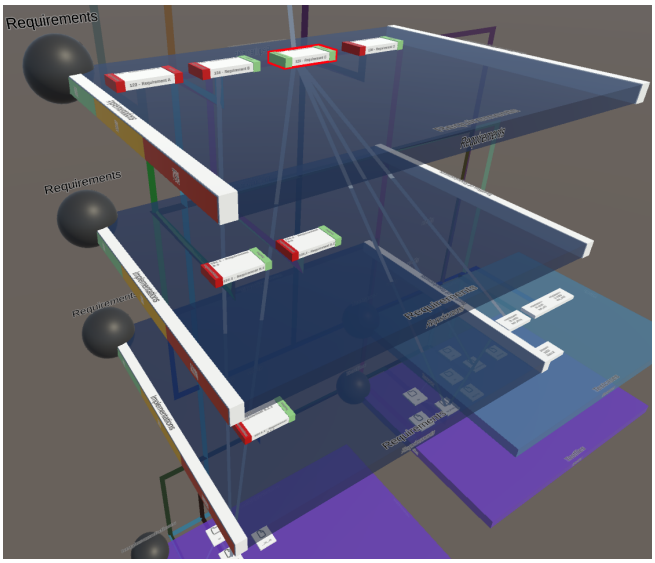


Figure 39. Requirements layers (one element selected showing direct spotlight relations).

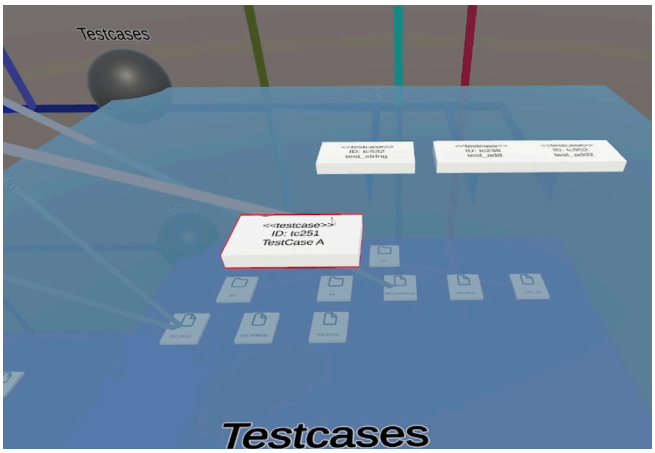


Figure 40. Test cases layer (one element selected).

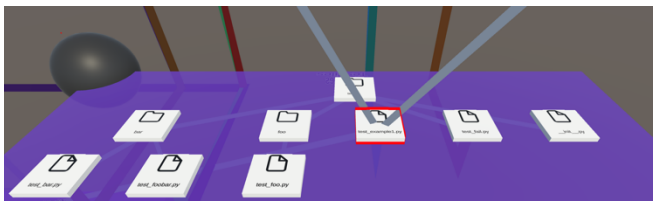


Figure 41. Test files layer showing tree (faintly in grey) of folders and files (with one element selected).

Requirements elements are colored on the edges (see Figure 42): the left side for implementation status and the right side for test case implementation status. The status is colored in three segments: red indicates the percentage of a (product or test) implementation missing, green the percentage completed, and yellow partial fulfillment. Thus, from the perspective of one side of all layers one can get a quick impression of the overall requirements implementation status or from the right side the test implementation status on a per-layer basis.

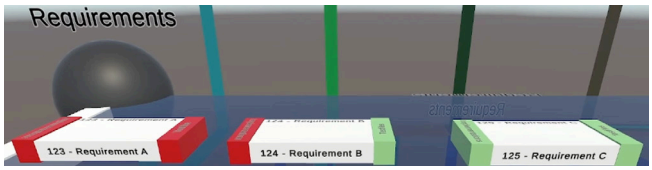


Figure 42. Use case layer.

The requirements layer shows a total fulfillment degree on the diagram edges: left for implementations and right for test cases. For example, in Figure 43 on that requirements layer three requirements are unimplemented while one is, so on the left edge 25% is shown for green and 75% in red, while in Figure 44 since three out of four requirements have tests, on the right edge red shows 25% and green shows 75%. This might be the case if the test team is ready with tests before the implementation has made progress, e.g., as with Acceptance Test-Driven Development (ATDD).

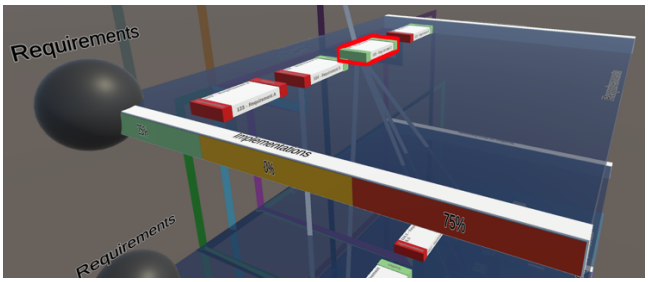


Figure 43. Requirements layer left side (implementation fulfillment).

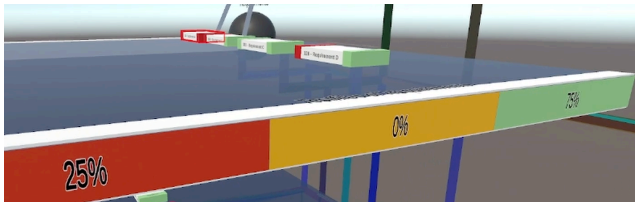


Figure 44. Requirements layer right side (test fulfillment).

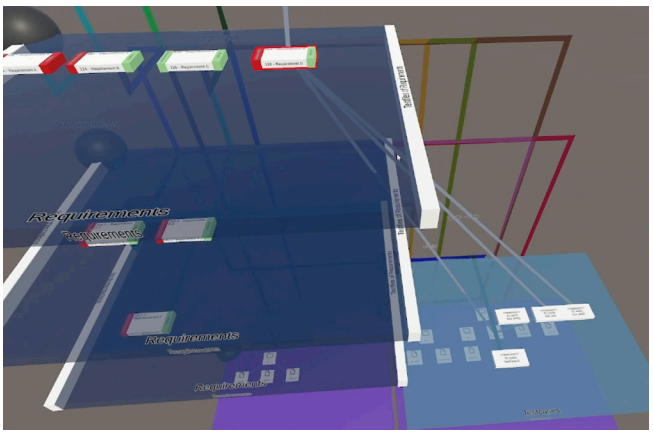


Figure 45. Selected requirement highlights the verifying test cases.

Figure 45 shows the traceability between a requirement to its associated verification test cases.

As to the VR-Tablet, additional detailed information about an object is shown. In Figure 46 for the selected requirement ID 126 the name (Requirement D) is shown, the number of test files (1) that exist, the test cases covered (2), and that no implementation was found. Also note no downward trace to the left towards the implementation is shown, only to the right towards tests. In contrast, Figure 47 shows test files and implementations exist for ID 125 (with downward traces in both directions), which is why both its edges are green, while Figure 48 shows a requirement with neither tests nor implementation.

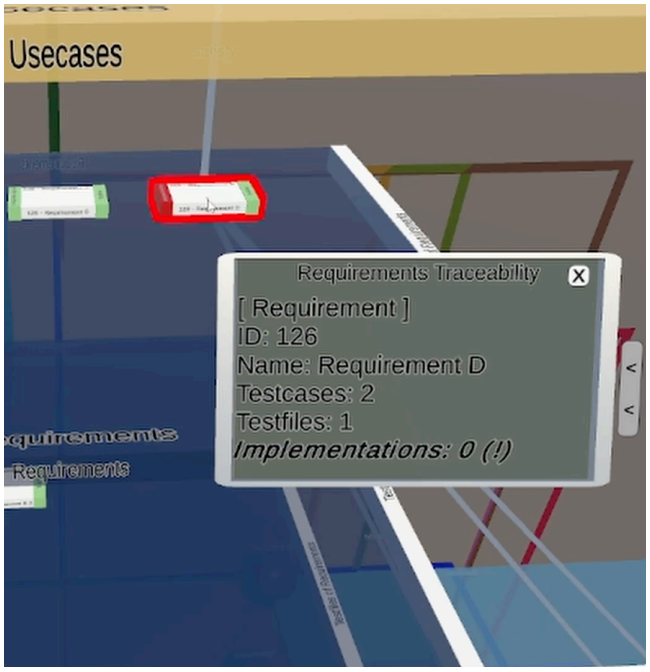


Figure 46. VR-Tablet shows detailed information about selected object.

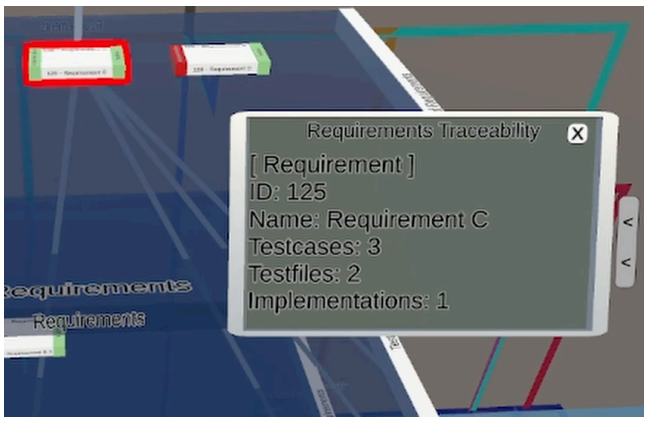


Figure 47. VR-Tablet shows detailed information about selected object.

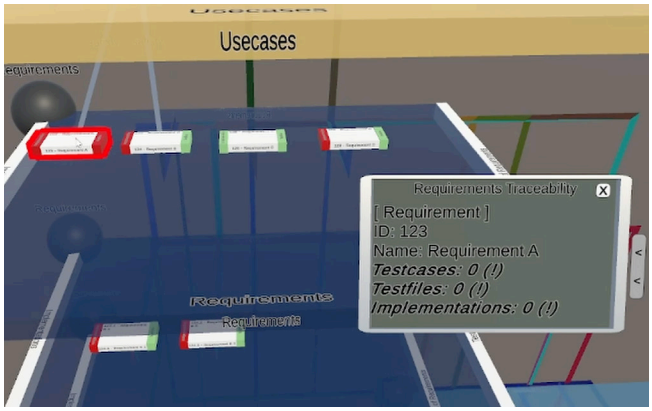


Figure 48. VR-Tablet shows detailed information about selected object.

To determine use case satisfaction, the VR-Tablet indicates if a selected use case is satisfied (Figure 49) or not (Figure 50).

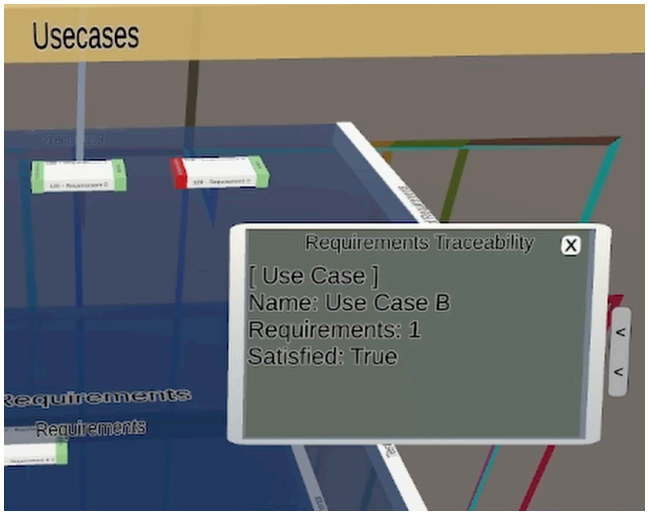


Figure 49. VR-Tablet shows selected use case satisfied since requirement implemented.

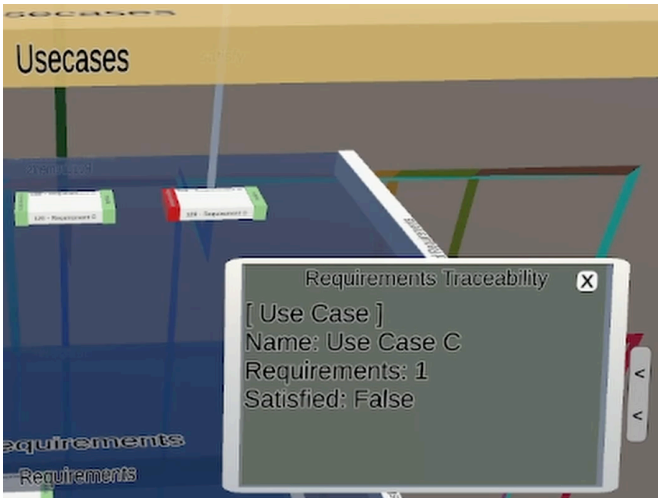


Figure 50. VR-Tablet shows selected use case unsatisfied since requirement not fulfilled.

When viewing test files, the VR-Tablet indicates the number of test targets that exists (the target can be found by following the trace), and if the test passed (Figure 51) or failed (Figure 52).

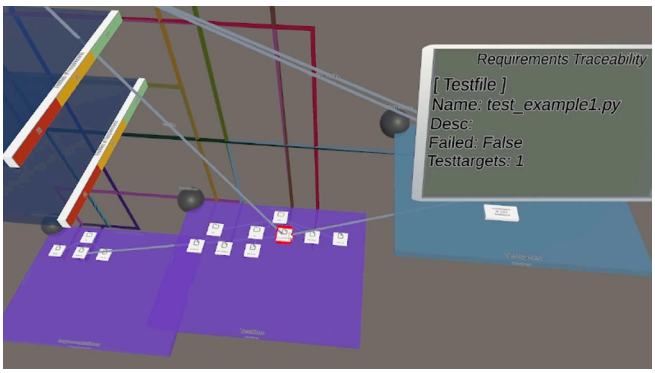


Figure 51. VR-Tablet shows associated test implementation name as test_example1.py with a single test target.

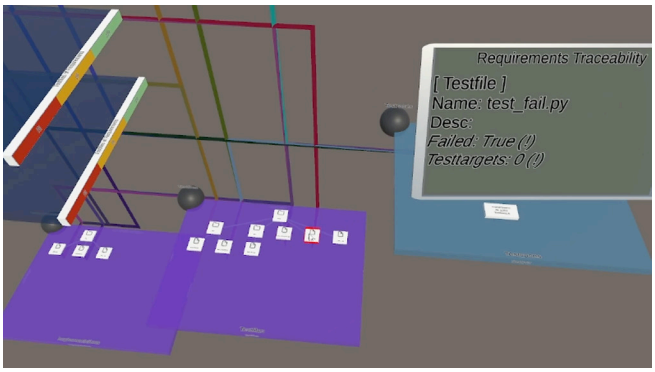


Figure 52. VR-Tablet showing test case status Failed as “True” and this test case having no associated test target information.

C. Discussion

The case-based evaluation of VR-SyML showed its ability to depict the various SysML diagram types in VR. Furthermore, the backplane supports the ability to quickly find the same elements across the various diagram types. As systems grow in complexity, these permits one to quickly find an element of interest in its various contexts. VR-SysML enables an immersive experience in the model, with the unlimited VR space supporting for larger models, multiple diagrams, prior or legacy model versions for comparison, and even heterogenous models displayed simultaneously.

Our traceability evaluation showed the code analyzer was able to automatically scan source code, test files and SysML diagram files and automatically ascertain traceability relevant information. The evaluation showed the ability of VR to provide an intuitive way to simply portray (via layers, the backplane, spotlights, and the VR-Tablet) the relevant artifacts and trace the relations between requirements, tests, and implementation to support V&V activities. This allows independent stakeholders who may not be system experts to evaluate the implementation and testing fulfillment without being inundated with perhaps irrelevant details in typical developer documentation.

VI. Conclusion

VR-SysML+Traceability contributes a VR-based solution concept for visualizing and interacting with SysML while adding automated requirements traceability and test tracing capabilities to support SW V&V. An immersive SysML model experience is provided for visually depicting and navigating SysML diagrams of models in VR. The solution concept was described, a VR prototype realized, while an evaluation using case studies showed its capabilities. Using VR hyperplanes, SysML diagrams are enhanced with 3D depth, color, and automatically-generated inter-diagram element followers based on our back-plane concept. Interaction is supported via a virtual tablet and keyboard. The unlimited space in VR facilitates the depiction and visual navigation of large models, while relations within and between elements, diagrams, and models can be analyzed. Furthermore, additional related (SysML or non-SysML) models can be visualized and analyzed simultaneously in VR, benefiting complex systems-of-systems architectures or collaboration. The sensory immersion of VR can support task focus during model comprehension and increase modeling enjoyment, while limiting the visual distractions that typical 2D display surroundings incur. The semi-automated traceability capability enhances the V&V opportunities by creating a simple and intuitive way to navigate and readily determine the degree of implementation and test progress and requirement fulfillment, even for quality assurance personnel who are not deeply acquainted with the project.

Future work includes support for creating models directly in VR, integrating further SysML tooling and simulation capabilities, supporting tighter and more comprehensive model verification and validation within the SysML diagrams, and conducting a comprehensive empirical study to evaluate

usability with various stakeholder groups in collaborative situations.

ACKNOWLEDGMENT

The author would like to thank Shadrach Arulrajah, Marie Bähre, and Christian Greiner for their assistance with the design, implementation, figures, and evaluation.

REFERENCES

[1] R. Oberhauser, “VR-SysML: SysML Model Visualization and Immersion in Virtual Reality,” International Conference of Modern Systems Engineering Solutions (MODERN SYSTEMS 2022), IARIA, 2022, pp. 59-64.

[2] OMG, “OMG Systems Modeling Language Version 1.6”, Object Management Group, 2019.

[3] R. Oberhauser, R., “VR-UML: The unified modeling language in virtual reality – an immersive modeling experience,” International Symposium on Business Modeling and Software Design (BMSD 2021), Springer, Cham, 2021, pp. 40-58.

[4] R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Shishkov, B. (ed.) BMSD 2019. LNBIP, vol. 356, Springer, Cham, 2019, pp. 170–187.

[5] C. Nigischer and D. Gerhard, “Lightweight visualization of SysML models in PDM systems,” DS 87-3 Proc.eedings of the 21st International Conf. on Engineering Design (ICED 17) Vol 3: Product, Services and Systems Design, 2017, pp. 61-70.

[6] I. Barosan, A. Basmenj, S. Chouhan, D. Manrique, “Development of a Virtual Simulation Environment and a Digital Twin of an Autonomous Driving Truck for a Distribution Center,” Software Architecture (ECSA 2020). CCIS, vol 1269, Springer, Cham, 2020, pp. 542–557.

[7] A. Mahboob, S. Husung, C. Weber, A. Liebal, and H. Krömker, “SYSML behaviour models for description of Virtual Reality environments for early evaluation of a product,” In DS 92: Proc. 15th Int’l Design Conf. (DESIGN), 2018, pp. 2903-2912.

[8] M. Ozkaya, “Are the UML modelling tools powerful enough for practitioners? A literature review,” IET Software, vol. 13, 2019, pp. 338-354. https://doi.org/10.1049/iet-sen.2018.5409

[9] M. Ozkaya and F. Erata, “A survey on the practical use of UML for different software architecture viewpoints,” Information and Software Technology, Vol. 121, 106275, 2020.

[10] P. McIntosh, “X3D-UML: user-centered design, implementation and evaluation of 3D UML using X3D,” Ph.D. dissertation, RMIT University, 2009.

[11] A. Krolovitsch and L. Nilsson, “3D Visualization for Model Comprehension: A Case Study Conducted at Ericsson AB,” University of Gothenburg, Sweden, 2009.

[12] C.S.C. Rodrigues, C.M. Werner, and L. Landau, “VisAr3D: an innovative 3D visualization of UML models,” In 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), IEEE, 2016, pp. 451-460.

[13] Y. Li and W. Maalej, “Which Traceability Visualization Is Suitable in This Context? A Comparative Study,” In: Regnell, B., Damian, D. (eds) Requirements Engineering: Foundation for Software Quality (REFSQ 2012), LNCS, vol 7195. Springer, Berlin, Heidelberg, 2012. https://doi.org/10.1007/978-3-642-28714-5_17

[14] Z. S. H. Abad, M. Noaeen and G. Ruhe, "Requirements Engineering Visualization: A Systematic Literature Review," 2016 IEEE 24th International Requirements Engineering Conference (RE), Beijing, China, 2016, pp. 6-15, doi: 10.1109/RE.2016.61

[15] A.A. Madaki and W.M.N.W. Zainon, “A Review on Tools and Techniques for Visualizing Software Requirement Traceability,” In: Mahyuddin, N.M., Mat Noor, N.R., Mat Sakim, H.A. (eds) Proceedings of the 11th International Conference on Robotics, Vision, Signal Processing and Power Applications, Lecture Notes in Electrical Engineering, vol 829, Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-8129-5_7.

[16] J. Vincur, P. Navrat, and I. Polasek, “VR City: Software analysis in virtual reality environment,” In 2017 IEEE international conference on software quality, reliability and security companion (QRS-C), IEEE, 2017, pp. 509-516.

[17] R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36.

[18] R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, Springer, Cham, 2018, pp. 83–97. https://doi.org/10.1007/978-3-319-94214-8_6

[19] R. Oberhauser, “VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality,” International Conference on Information, Process, and Knowledge Management (eKNOW 2022), IARIA, 2022, pp. 29-36.

[20] R. Oberhauser, P. Sousa, and F. Michel, "VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality," In: Shishkov B. (eds) Business Modeling and Software Design. BMSD 2020. LNBIP, vol 391, Springer, Cham, 2020, pp. 221-239. doi: 10.1007/978-3-030-52306-0_14

[21] R. Oberhauser, M. Baehre, and P. Sousa, “VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality,” In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2022. LNBIP, vol 453, pp. 122-140. Springer, Cham. doi:10.1007/978-3-031-11510-3_8

[22] R. Oberhauser, "VR-TestCoverage: Test Coverage Visualization and Immersion in Virtual Reality," The Fourteenth International Conference on Advances in System Testing and Validation Lifecycle (VALID 2022), IARIA, 2022, pp. 1-6.

[23] R. Oberhauser, “VR-Git: Git Repository Visualization and Immersion in Virtual Reality,” The Seventeenth International Conference on Software Engineering Advances (ICSEA 2022), IARIA, 2022. To be published.

[24] E.M. Reingold and J.S. Tilford, “Tidier drawings of trees,” IEEE Transactions on software Engineering, (2), 1981, pp.223-228.

[25] A.R. Hevner, S.T. March, J. Park, and S. Ram, “Design science in information systems research,” MIS Quarterly, 28(1), 2004, pp. 75-105.

Hand Gesture Recognition System for the Physical Search System

1st Shin Kajihara
Graduate School of Science and Engineering
Saga University
Saga, Japan
email: 20634002@edu.cc.saga-u.ac.jp

2nd Masato Okazaki
Graduate School of Science and Engineering
Saga University
Saga, Japan
email: 22726005@edu.cc.saga-u.ac.jp

3rd Chika Oshima
Faculty of Science and Engineering
Saga University
Saga, Japan
email: sj5872@edu.cc.saga-u.ac.jp

4th Koichi Nakayama
Faculty of Science and Engineering
Saga University
Saga, Japan
email: knakayama@is.saga-u.ac.jp

***Abstract*—In this paper, we proposed a hand gesture recognition system for searching for lost objects using a physical search system (PSS). The PSS detects all displaced objects in a physical space using two cameras and a computer based on image differences-detection technology. When users tell the PSS what the lost object is, using hand gestures to describe it, such as its size and location, may be useful, as may words that describe the object's name, color, and the time it was last seen. The hand gesture recognition system was developed and experiments were conducted to examine how accurately the system can estimate the size indicated by the width between the user's hands. Also, to allow users to register various gestures as the commands they want to use, we investigated the recognition rate of finger gestures. As a result, the system could measure the width between users' hands with almost no errors, based only on the image taken by the camera and a marker. Moreover, the finger gestures could be recognized with high accuracy, unless it was difficult for users to reproduce the gestures that had been pre-registered. In the PSS, the displaced object's images are grouped into clusters that contain the same objects' images and data about their features. When a user tells the PSS the features of what they want to find, using their hand gestures, the PSS can present to the user images of the object in an appropriate folder (cluster) that matches the request. Finally, once the user identifies the lost object's image, the PSS displays where and when the object was last seen/lost.**

***Index Terms*—Hand gesture; physical search system; MediaPipe.**

I. Introduction

Keyword and image searches are often used to search for data online. By contrast, the physical search system (PSS) [1], [2] looks for objects in physical space without requiring any sensors, other than a camera or data for pre-learning, and enables the retrieval of any object that has moved within a given physical space.

We sometimes use hand gestures when telling someone about a lost object; "a board about this size" with our hands outstretched, or "the remote control was over there," while pointing with the index finger. Even when using the PSS, it is desirable to be able to input information about the object's size and an approximate location in physical space using hand gestures.

Hand gesture recognition can be broadly divided into wearable and non-wearable types. In wearable types, there are an acceleration sensor [3], [4] and optical markers, such as color and reflective markers [5], [6]. However, for daily use, it is inconvenient to wear devices and markers on the hands and fingertips.

In non-wearable types, Leap Motion [7] can recognize hands and fingers by irradiating infrared rays with a small device, but they can only be detected up to a distance of about 0.5m from the device. OpenPose [8] can acquire the position and posture of fingers only based on camera images. With a high-resolution zoom camera, even far-distant hand gestures can be recognized, but the positions of photographed fingers can only be acquired two-dimensionally. Users do not always make gestures toward the camera because they tend to point in the direction where they think the lost object is, or they express the shape of the object in three dimensions. Therefore, a hand gesture recognition system needs to work not only in a non-wearable format, but also to acquire three-dimensional (3D) positions in physical space (hereinafter "the world coordinate system").

MediaPipe Hands [9], [10], a non-wearable type, acquires the estimated Z coordinate, in addition to 2D coordinates (X, Y) from a camera image. In this paper, we propose a system that can indicate features of a lost object based on hand gestures. Users can command the system with the hand gestures they determine to search for the lost object with the PSS, such as indicating a size of the lost object, pointing to the approximate location of the object before it was lost. No sensors, other than a camera in a real space and MediaPipe, are required.

In the next section, the PSS is introduced; the experiment's results are explained in Section II. Then, in Section III, a hand gesture recognition system is proposed. Two experiments are

conducted to examine how accurately the system can estimate the length between a user's hands and interpret their finger gestures. The paper concludes in Section IV.

II. Physical Search System (PSS)

A. Overview

This section explains the overall structure of the proposed PSS [1] [2]. Figure 1 shows the PSS' hardware configuration. The PSS consists of two cameras that constantly capture the target area and a computer that processes the images photographed by the cameras. The PSS' software configuration consists of a displaced object detection unit that extracts the displaced objects from images photographed by each camera, a displaced object image-clustering unit that creates clusters, and a search results display unit that retrieves and displays the displaced objects.

In the displaced object detection unit [2], the photographed images are processed in the order in which they are photographed. The image at a certain time is then compared at the pixel level with the image photographed at a previous time. When a pixel with a difference of a certain standard or more is detected, it is determined that something has been displaced. The PSS can also detect people, and the photographs can define areas in which no one or nothing is present. In other words, the PSS does not yet detect objects that are moving/rotating around of the center of gravity, but it can detect displaced objects by comparing sets of images.

When objects overlap, we can obtain expected results, if they are displaced in order. For example, Object A is placed at a certain place. After the system has photographed an image near Object A, Object B is placed on top of Object A. If the PSS photographs the place again before each object is moved, both Objects A and B will be detected correctly. When two overlapping objects move together, Objects A and B are detected as a single object, so if Object A is pulled out from under Object B, Object A will not be detected. However, if Object A is placed elsewhere, it will be detected as a displaced object.

Figure 2 shows the differences between the two images in white. The target area is cropped as a rectangle [11]. The cropped image is called a "displaced object image." As described in Section II-B, in the displaced object images clustering unit, the displaced objects' images are grouped into clusters that contain the images of the same objects in each location and stored in the PSS. In the search result display unit, as shown in Figure 3, when a PSS user searches for a lost object (a displaced object), the search results are displayed in an application that displays augmented reality (AR) using an AR marker and an AR display terminal [2].

B. Two-step Feature Clustering Algorithm

This section describes a two-step feature clustering algorithm (TFA [2]). At first, the displaced object images are processed with the x-means clustering algorithm [18]. Then, the PSS user manually deletes a few folders (clusters) in

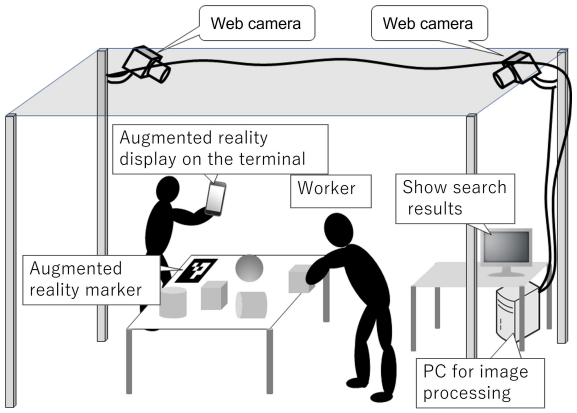


Fig. 1. Construction of the PSS hardware [2].

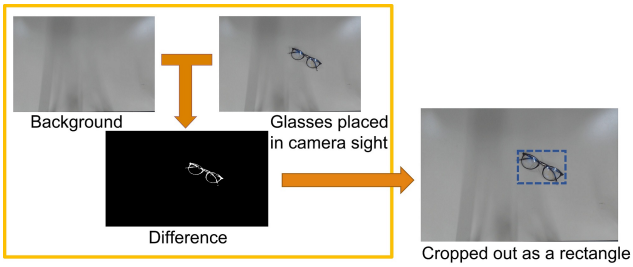


Fig. 2. How to crop out a displaced object [11].

which only noisy images are included. The reason why x-means was employed is that x-means is a method of clustering while automatically estimating the number of clusters k of k-means [19]. Therefore, the x-means clustering is a type of unsupervised learning like the k-means, wherein the data points (the features of the displaced object images) are grouped into different clusters based on their degree of similarity.

Next, a method of generating displaced object images with the feature are explained. ResNet50 [12], [13] is applied to the displaced object images to quantify their features. ResNet50 is a convolutional neural network that is pre-trained on ImageNet [14], an image database. Therefore, the user does not need to prepare any image-learning data.

ResNet is a residual network designed to alleviate the vanishing/exploding gradient problem caused by stacking residual

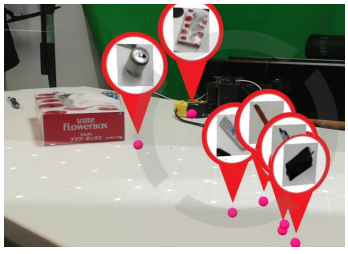


Fig. 3. Icons displays where and when the object was last seen/lost [2].

blocks. In ResNet-50, one residual block consists of three convolution layers. The size of the convolution kernel, which is the element of convolution operation in the convolutional layer, should be smaller than the size of the input image [15]. The stacked layers in the residual blocks have $1 \times 1, 3 \times 3$, and 1×1 convolution layers. The 1×1 convolution first reduces the dimensions. In the next layer, the bottleneck 3×3 layer, the features of the images are calculated. Then, the dimension of depth is again added in the next 1×1 layer (bottleneck) [16]. The final convolutional layer outputs 2048 feature maps of size 7×7.

In the PSS, the images of the displaced object are resized to 224 $\times$ 224 pixels and entered into ResNet50. Then, each resized image is flattened into the 100352-dimentional vector ($1 \times 7 \times 7 \times 2048$, which is a tensor: $depth(none) \times width \times height \times channel$) [17]. We call these "image features" in this paper.

Then, the displaced object images with the feature are processed with the x-means clustering algorithm [18]. A cluster number is assigned to each cluster, as determined by the x-means method. All displaced object images are stored in folders according to their cluster number.

There are a few folders (clusters) in which only noisy images are included. The PSS user manually deletes these. Then, the same processing protocol as used in the first stage is performed again for all images in the remaining clusters (the second step).

However, some noisy images will remain in a few folders [2], so there is room for improvement in accuracy. Therefore, a linking method (LM) was proposed to improve the accuracy of the TFA clustering [1].

C. Linking Method

This section describes the LM [1]. LM eliminates noisy images by creating pairs of images of highly similar displaced objects based on photographs taken simultaneously by two cameras [1].

In the PSS [2], two cameras (Cameras A and B) usually take pictures of the same displaced object at the same time from different angles. However, noisy images are photographed by only one of the two cameras, because noisy images are a result of misrecognition due to light rays or mistakes in cropping out the object parts of the images. Therefore, as shown in Figure 4, in the LM, pairs of the displaced object images with high degrees of similarity are created from the displaced object images derived from the photographs taken simultaneously by Cameras A and B. This process is called "linking" in this paper. In other words, a pair combination is created with a displaced object image derived from Camera A's photograph and another displaced object image derived from Camera B's photograph. Displaced object images obtained from only one of the cameras cannot be paired.

Next, the method for calculating the similarity between the displaced object images is explained. "imgsim [20]" is a library for computing perceptual hashes of images. The "distance" between images can be calculated using the imgsim library. The distances between the displaced object image, "a1," derived from Camera A's photograph and the displaced object images, "b1 to bx," derived from Camera B's photograph, are calculated. The higher the degree of image similarity, the smaller the distance between them. The distance between identical images is 0.

As shown in Figure 5, pairs are created in order, starting from those with the smallest distance value (the highest degree of similarity) between two images. For example, when two displaced object images are obtained from Cameras A and B's photographs taken at a certain time, there are four possible pair combinations. The image that is paired with another image is excluded from the candidate images for the other pairs. In addition, combinations with distance values exceeding 23 are not considered pairs. A gathering of the pairs is called a "pair group."

D. Brush up TFA Clustering Results with Pair Groups Created Using LM

This section explains a method for combining the TFA and the LM. Figure 6 shows that the displaced object images are updated by comparing the TFA results with that of LM; then, the clusters (folders) are reorganized. The displaced object images in the clusters that do not overlap with the displaced object images of the pair group are deleted. In this process, the noisy images and the images for which one camera has failed to detect a displaced object can be deleted from the folders.

Finally, the clusters created by TFA are reorganized. If two displaced object images that are paired belong to different clusters, they are processed as follows: the similarity (distance) between each of two images and other images that belong to the same cluster of each of two images is calculated using the imgsim library. Next, the averages of the distances in each cluster are calculated. The one with the larger average value moves to the cluster that includes the other displaced object image with the smaller average value. For example, Image_p, which belongs to Cluster_P, is paired with Image_q, which belongs to Cluster_Q. The distance values are calculated between Image_p and each of the other images in Cluster_P, and between Image_q and each of the other images in Cluster_Q. Then, the averages of the distance values are calculated for both Cluster_P and Cluster_Q. If the average of distances between Image_p and the other images in Cluster_P is larger than that of Cluster_Q, Image_p is moved to Cluster_Q.

E. Experiments for the usefulness of LM

1) Aim: In this section, we detail an experiment conducted to compare the accuracy of clustering between the combination of LM with TFA and TFA alone [1].

2) Method: Figure 7 shows ten objects on a table. The objects were a red pen, a green pen, a smartphone tripod, a box of tissues, a cup of coffee, a black smartphone, a box of darts, a dart, a plastic bag of replacement dart feathers, and gum tape. Two cameras were located so that the entire table could be photographed from two different directions. Even if

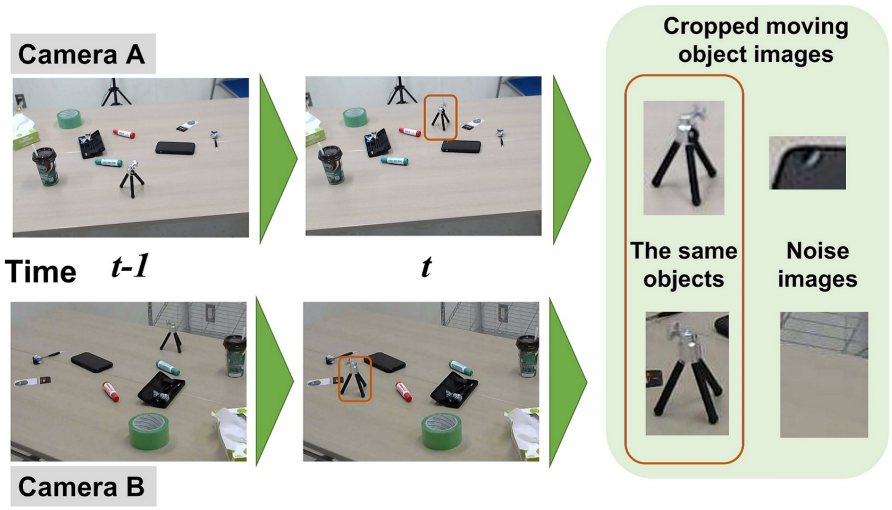


Fig. 4. Noisy images cannot be paired with another image.

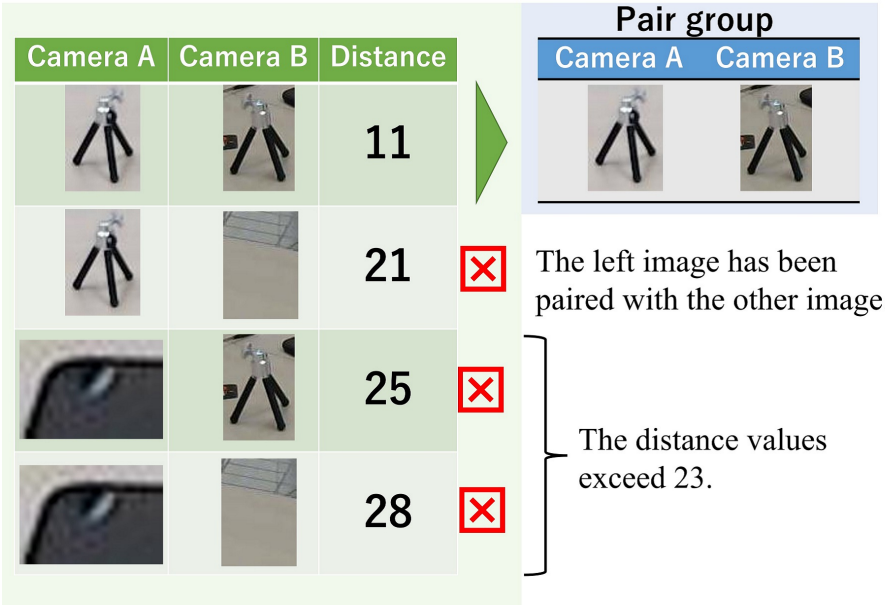


Fig. 5. Create a pair based on the similarity between two images.

the PSS is running, when there are people present, nothing will not be photographed.

During the experiment, one of the authors moved one of the objects on the table and then moved beyond the ranges of the cameras. After confirming that the PSS recognized the displaced object, they moved the next object on the table. This method was applied to the ten objects. They moved each object 10 times in two conditions, “LM with TFA” and “TFA.”

This process was repeated twice on two different tables in two different rooms, Rooms C and D. In Room C, the ten objects were on a desk, as shown in Figure 8. This is a dimly lit space because three displays are lined up, and the fluorescent lamp is not directly overhead. In Room D, a large table was placed in the center of the room, with fluorescent lights directly above it. There was nothing around it to block the light, as shown in Figure 9.

The PSS created clusters in both conditions. The accuracy of the clustering is indicated in precision values, recall values, and F-measures. All displaced object images showing one of the ten objects are regarded as an “actual positive.” The cluster in which the most images is included is considered to be a correct cluster, and the displaced object images of the correct cluster are regarded as a “predicted positive.” In the predicted positive images, the actual positive images are considered to be true positives (TP), and the others are considered false positives (FP). In the actual positive images, the images that

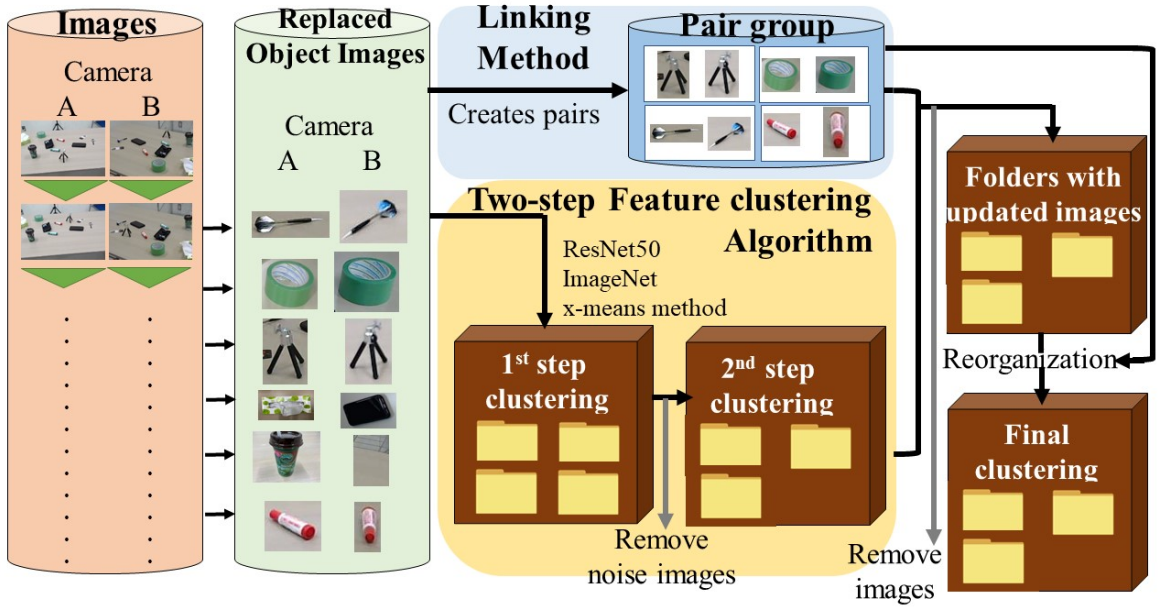


Fig. 6. Combine the results of the linking method (LM) with the two-step feature clustering algorithm (TFA) to update the images; then, reorganize the clustering.

are not in the correct cluster are considered false negatives (FN).

The recall is calculated using the following formula.

$$Recall = \frac{TP}{TP + FN} \quad (1)$$

The precision is calculated using the following formula.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

The F-measure is calculated using the following formula. The F-measure represents the harmonic mean of precision and recall.

$$F - measure = \frac{2Precision * Recall}{Precision + Recall} \quad (3)$$

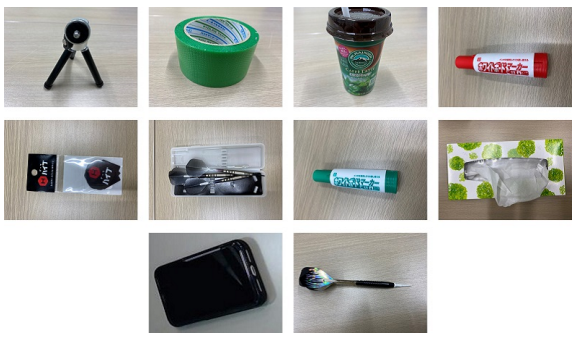


Fig. 7. Ten kinds of objects for the experiment.

Fig. 8. Desk in Room C.

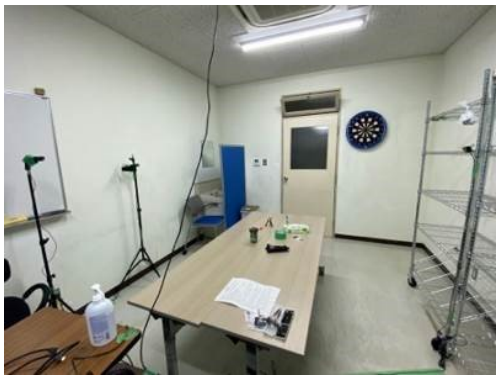


Fig. 9. Table in Room D.

Fig. 10. Images that missed a part of the gum tape.

3) Results:

Comparison of the Results of TFA and LM with TFA

The PSS created 11 clusters in both conditions. Each displaced object image was grouped into one to five clusters, depending on the type of displaced object. For example, all images of the smartphone tripod were grouped into Cluster_3 in the LM with TFA condition. The images of the green pen were grouped into five kinds of clusters in the TFA condition.

Table I shows the recall values, precision values, and F-measures to compare the results of the two conditions in both Rooms C and D [1]. For eight out of ten objects, the recall values were higher in the LM with TFA condition than in the TFA alone condition. In particular, the recall value of the gum tape under the LM with TFA condition was improved by 17%, compared to the TFA alone condition, and it became even closer to 100%.

For all objects, the F-measures were higher in the LM with TFA condition than in the TFA alone condition. However, the precision values of the LM with TFA condition were almost the same as that of the TFA alone condition. The displaced object images, the smartphone, the box of darts, and the dart remained around 30%.

Comparison of the Results of Rooms C and D

Table II shows the precision values, recall values, and F-measures to compare the results of Rooms C and D [1]. The number of images is less than Table I because of the results for each room. Therefore, the number of clusters was different, and these evaluated values between Tables I and II are different. The F-measures of all displaced object images in Room D were higher than that of Room C.

F. Discussion

The results showed that LM with TFA improved the clustering over TFA alone. As an example of improvement, the images of the gum tape were grouped into four clusters in the TFA alone condition because there were some images that missed a part of the gum tape, as shown in Figure 10. However, in the LM with TFA condition, most images of gum tape could be grouped into one cluster.

The F-measures for the red and green pens were not high, even in the LM with TFA condition. Since the shapes of these pens are similar, it was difficult to group them into one cluster using these algorithms. An algorithm using color features should be applied to such objects [2].

For the smartphone tripod and the cup of coffee, the precision values in the LM with TFA condition were lower than in the TFA alone condition. In the LM process, contrary to our expectations, the images of the smartphone tripod and the cup of coffee were paired with noisy images. Something similar to these objects was photographed as noisy images.

There were differences in accuracy depending on the room. The table in Room D was brighter than the desk in Room C. It can be suggested that the brighter space led to higher accuracy in detecting the displaced objects.

III. Hand Gesture Recognition System

A. Overview

In this section, a hand gesture recognition system is proposed. It consists of a registration mode, which corresponds to the initial settings when starting to use the system, and a recognition mode, which recognizes hand gestures by actually using the system. Section III-B describes the camera registration phase and its coordinate system, while Section III-C describes the gesture registration phase for defining the gestures to be discriminated. Section III-D describes the position acquisition phase, which obtains finger positions, and Section III-E describes the gesture recognition phase, which determines to which gesture the shape created by fingers corresponds.

B. Registration Mode: Camera Registration Phase

An augmented reality marker (AR marker) printed in an arbitrary size is placed at an arbitrary location that can be seen by all cameras. The experiment detailed in this paper used an ArUco marker [21] printed in a 570 mm $\times$ 570 mm square, as shown in Figure 11. The center of the marker was set as the origin of the world coordinate system. Each camera reads the marker and then calculates and stores its own position and orientation (camera parameters) in the world coordinate system.

C. Registration Mode: Gesture Registration Phase

In this phase, gestures to be discriminated are registered with the proposed system. The position and orientation of each finger are estimated by MediaPipe Hands [9], [10]. Figure 12 shows the numbered coordinates of 21 parts of each finger that MediaPipe Hands can acquire from each camera based on its images [10]. The coordinates are two-dimensional (X, Y) in the camera image, and the depth coordinates (Z) are estimated by MediaPipe Hands. Since the video is processed as time-series data at 10 frames per second, the coordinates of the hand gesture video for 1 second are registered in the system as real values of 21 points $\times$ 3 axes (X, Y, Z) $\times$ 10 frames.

TABLE I
ACCURACY OF CLUSTERING IN THE CONDITIONS TFA ALONE AND LM WITH TFA.

	Recall		Precision		F-measure	
Displaced objects	*TFA*	*LM with TFA*	*TFA*	*LM with TFA*	*TFA*	*LM with TFA*
Red pen	0.50	0.53	0.63	0.64	0.56	0.58
Green pen	0.38	0.43	0.47	0.43	0.42	0.43
Tripod	0.93	1.00	1.00	0.95	0.96	0.98
Box of tissues	0.49	0.63	1.00	1.00	0.65	0.77
Cup of coffee	0.91	1.00	1.00	0.94	0.95	0.97
Smartphone	0.42	0.58	0.23	0.29	0.30	0.39
Box of darts	0.44	0.69	0.30	0.30	0.36	0.42
Dart	0.75	0.75	0.24	0.27	0.36	0.40
Plastic bag	0.66	0.63	0.48	0.57	0.56	0.60
Gum tape	0.76	0.93	1.00	1.00	0.86	0.96
Average	0.62	0.72	0.64	0.64	0.60	0.65

TABLE II
ACCURACY OF CLUSTERING IN THE CONDITIONS OF ROOMS C AND D.

	Recall				Precision				F-measure			
	TFA		*LM with TFA*		*TFA*		*LM with TFA*		*TFA*		*LM with TFA*	
Displaced objects	C	D	C	D	C	D	C	D	C	D	C	D
Red pen	1.00	0.88	1.00	0.75	0.43	0.45	0.43	0.55	0.60	0.60	0.60	0.63
Green pen	0.87	0.81	0.95	0.63	0.48	0.81	0.50	0.45	0.62	0.55	0.66	0.53
Tripod	0.76	1.00	0.90	1.00	0.90	0.90	1.00	0.90	0.83	0.95	0.95	0.95
Box of tissues	0.50	1.00	0.50	1.00	0.50	0.80	1.00	1.00	0.50	0.89	0.67	1.00
Cup of coffee	0.93	1.00	0.71	1.00	0.87	0.86	1.00	1.00	0.90	0.92	0.83	1.00
Smartphone	0.64	0.81	0.76	0.63	0.20	0.39	1.00	0.50	0.31	0.53	0.86	0.56
Box of darts	0.53	0.81	0.47	0.63	0.22	0.39	0.57	0.48	0.31	0.53	0.52	0.63
Dart	0.69	0.56	1.00	1.00	0.09	0.90	0.12	0.95	0.17	0.69	0.22	0.97
Plastic bag	1.00	0.80	0.95	0.93	0.82	0.92	1.00	0.93	0.90	0.86	0.98	0.93
Gum tape	0.78	1.00	0.78	1.00	1.00	0.95	1.00	1.00	0.88	0.97	0.88	1.00
Average	0.77	0.87	0.80	0.86	0.55	0.74	0.76	0.78	0.60	0.75	0.72	0.82

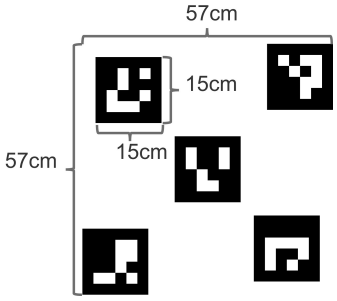


Fig. 11. A marker for determining the orientation and position of the camera on the world coordinate system.

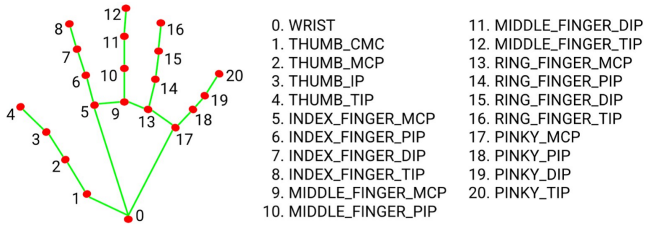


Fig. 12. 21 hand landmarks (quoted from [10]).

However, this coordinate is the value in the coordinate system for each camera. As shown in Section III-D, the coordinate values are converted to values in the world coordinate system based on the camera's unique parameters (see Section III-B).

D. Recognition Mode: Position Acquisition Phase

This section shows how to convert the coordinates of the hand in the camera's coordinate system to the world coordinate system. First, 0: WRIST (Figure 12) (hereinafter, "wrist coordinate") is used as the representative value of the hand position. When only one camera is used, there is a large error in the depth coordinate (Z-axis) in the camera coordinate system; therefore, the 3D position (coordinate) of the hand in the world coordinate system cannot be determined uniquely. With this system, however, since the camera parameters are obtained using AR markers, the position and orientation of each camera in the world coordinate system can be calculated. Then, since the positions of the hands are detected simultaneously with two cameras, their 3D positions in the world coordinate system can be calculated.

To explain in detail, the coordinates in the world coordinate system are obtained from the perspective projection transformation matrix, which indicates the position and orientation of the two cameras and the wrist coordinate in the screen obtained by each camera. The wrist coordinate is estimated to lie on a straight line passing through the camera in 3D space, calculated from the position and orientation of one camera and

the wrist coordinate estimated from the images taken by that camera. Similarly, based on the image of the other camera, the wrist coordinate is estimated to lie on another straight line passing through the camera in 3D space. The midpoint of the line segment representing the shortest distance from these two straight lines is the wrist coordinate in 3D space.

E. Recognition Mode: Gesture Detection Phase

The time-series data of each recognized finger position (hand gesture) is coordinate-transformed so that the positions and orientations match the registered hand gestures (see Section III-C). As shown Figure 13, the position and orientation of the hand gesture is transformed to overlap with the positions and orientations of the registered gestures at Positions 0, 5, and 17 (see Figure 12). Then, the similarity between the hand gesture captured by the camera and the registered hand gesture is determined. If the registered hand gesture has 10 frames, the similarity is continuously determined for the latest 10 frames of the obtained data. Three similarities are used: cosine similarity for position, Euclidean distance of position of each part, and velocity of wrist position. When each similarity is higher than the threshold value, it is determined that the corresponding hand gesture was performed.

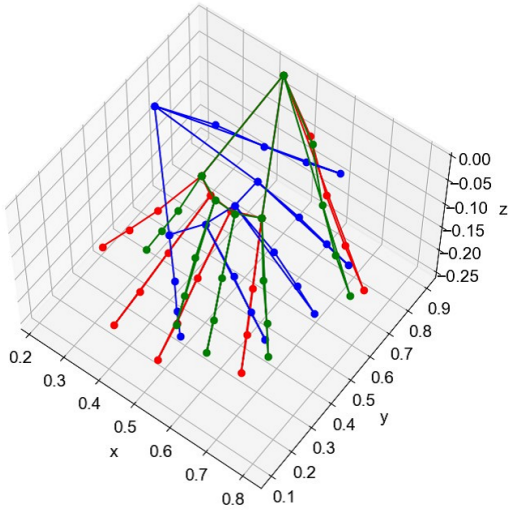


Fig. 13. The hand gesture is transformed to overlap with the registered gestures.

F. Evaluation of Size Expressions by Hand Gestures

1) Aim: This section verifies how accurately the system can estimate the size indicated by the user's hand gesture in real space.

2) Setting: As shown in Figure 14, four cameras [22] with a height of 2600 mm facing the direction of a square table were placed at the four corners of a square with a side of 3200 mm. The height of the table was 660 mm, and each side was 800 mm long. The ArUco marker [21] was placed in the center of the table as the origin of the world coordinate system.

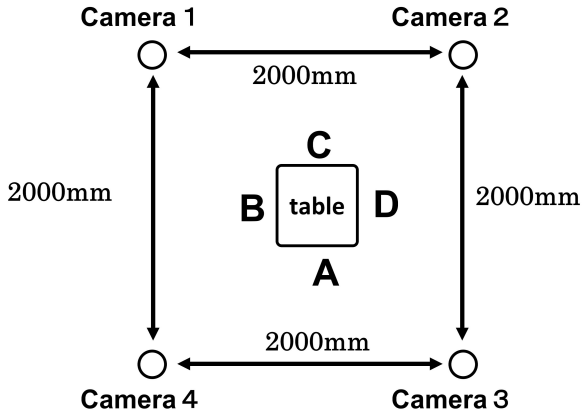


Fig. 14. A Table and Four Cameras.

3) Method: Participants were five males in their twenties. The space between their left and right hands was fixed using a wristband with a string attached. As shown in Figure 15, the wristbands were made of Velcro with 300-, 600-, or 900 mm-long strings. First, the participants separated their arms to the full length of the wristband string while they clenched their fists. Then, the participants turned their palms upwards. In this experiment, the system regarded the palm-up action as a gesture indicating length and measured the width between the hands.

The first author preregistered in the system these palm-up gestures with lengths of 300, 600, and 900 mm. Each participant stood facing the center of the table in an assigned position, A–D (see Figure 14). Then, the participant was instructed to perform the palm-up gesture with the 300 mm wrist band. After the length between their hands was recognized, they moved clockwise and performed the palm-up gesture with the same wrist band again. After they completed it for all four positions, they performed it with the remaining two length wristbands, again moving to each of the four positions. A total of 60 measured values (5 participants, 4 positions, 3 length wristbands) was acquired.

The absolute and relative errors for each measurement value were calculated as follows:

$$AE = |d_i - d| \tag{4}$$

$$RE = \frac{|d_i - d|}{d} \times 100 \tag{5}$$

where AE means the absolute error and RE means the relative error. d_i means the measurement value and d means the length of each wrist band (300, 600, or 900 mm).

Then, means, standard deviations (SD), and coefficients of variation (CV) were calculated for each condition (300, 600,

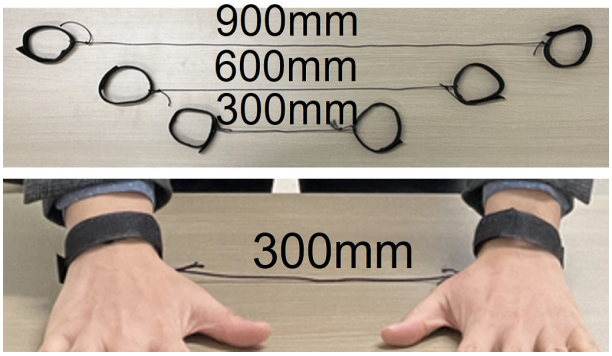


Fig. 15. Wristbands

distance of 300 mm, while the other had a horizontal distance of 600 mm from the desk.

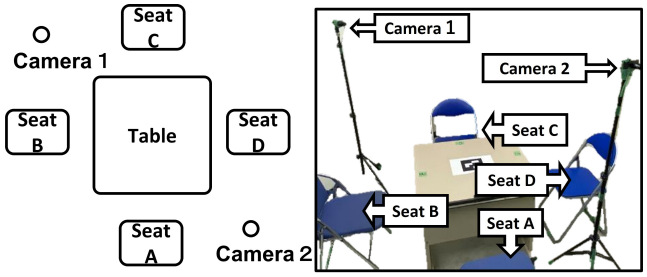


Fig. 16. Four Seats and Two Cameras Around the Table.

and 900 mm). Because the means were drastically different from one another, the CV was calculated as follows:

$$CV = \frac{s}{\bar{d}} \tag{6}$$

s and $\bar{d}$ indicate the standard deviation and means of the width measurements, respectively. CV is the ratio of the standard deviation to the mean.

4) Results: Table III shows the width measurements between the hands using the system, AE, and RE. The mean, SD, and CV are shown at the ends of the columns.

The means of AE were 14.5, 40.2, and 48.6 mm in each condition, respectively. The average error was less than 5 cm. The means of RE show that the error rate was smallest under the 300 mm condition.

By contrast, the CV results show that the measurement values while wearing the 300 mm wristband had the greatest variation among the three conditions (see Table III).

5) Discussion: In this section, the participants wore wristbands connected by 300, 600, and 900 mm strings on both hands. They were asked to make a gesture of fully spreading their hands. The error between the actual spreading length and the length measured by the system was investigated. Because the average errors were small, the results showed that it is possible to convey the length of a search object to the system using this hand gesture.

G. Recognition of Finger Gestures' Accuracy

1) Aim: In addition to hand gestures that indicate the size of objects, various gestures can be registered as commands that users want to use. In this section, 10 kinds of finger letters were used to examine the recognition rate of detailed finger gestures.

2) Setting: Figure 16 shows the setup for the experiment. Four seats (A–D) with different angles of 90 degrees were prepared around a square table. The height of each seat was 440 mm, the size of the seat was about 400 mm square, and the height of the highest point of the backrest was 800 mm from the floor. Two web cameras [22] were set on the diagonal extension of the desk. One of the cameras had a horizontal

3) Method: Participants were 11 males in their twenties who had never used finger gestures. Each participant sat in Seats A–D in turn. The participants were presented with a list of finger letters, as shown in Figure 17. The "a, i, u, e, o" are the Japanese vowels. The "ka, ki, ku, ke, ko" show five consonant and vowel combined pronunciations. The participants were asked to spell the following five words with their fingers: "a-ka," "i-ku," "u-ki," "o-ke," and "ko-e." These finger letters were preregistered in the system by the second author. However, these were not intended to be recognized as "characters." These finger letters were used to examine how recognizable the finger gestures were.

Each participant was seated facing the center of the table in an assigned seat, from A–D. Then, they was instructed to perform the first finger letter, a-ka, with his right hand until the camera recognized it. The orientation of the hands and fingers was arbitrary. If a gesture was not recognized after repeating it 10 times, it terminated as a recognition failure. After a-ka was recognized, they moved clockwise and performed the same finger gesture again. After completing this in all four seats, they performed the remaining four words' finger gestures, again, sitting in each of the four seats. A total of 220 types of finger gesture data (11 participants, 4 seats, 5 types of hand gestures) were acquired.

When the first author piloted the recognition of the finger letters using this method, the recognition rate was almost 100% for any gesture and any seat. The participants were asked to examine the list of finger letters at the beginning of the experiment and then performed the finger letters without prior practice.

4) Results: A total of 199 of the 220 finger letter trials (90.5%) were recognized in fewer than 10 trials. The average number of trials for 199 was 1.98 times.

Table IV shows the recognition rate and average number of attempts per finger gesture. The gestures for o-ke and i-ku had both a high recognition rate and a low average number of trials. O-ke moves from "o," with all fingers curled, to "e," with four fingers extended. I-ku moves from "i," with only the little finger upright and the others curled, to "ku," with four fingers extended horizontally.

TABLE III
WIDTH INDICATED BY HAND GESTURE AS MEASURED BY THE SYSTEM.

	String length								
	300mm			600mm			900mm		
Participant	Measured value(mm)	AE (mm)	RE (%)	Measured value(mm)	AE (mm)	RE (%)	Measured value(mm)	AE (mm)	RE (%)
1	275.4	24.6	8.2	617.1	17.1	2.8	895.3	4.7	0.5
1	286.9	13.1	4.4	603.1	3.1	0.51	889.6	10.4	1.2
1	282.6	17.4	5.8	618.8	18.8	3.14	960.4	60.4	6.7
1	286.8	13.2	4.4	650.7	50.7	8.45	957.7	57.7	6.4
2	316.7	16.7	5.6	659.4	59.4	9.9	990.5	90.5	10.1
2	329.3	29.3	9.8	647.4	47.4	7.9	994.9	94.9	10.6
2	317.2	17.2	5.8	629.7	29.7	5.0	998.5	98.5	11.0
2	317.1	17.1	5.7	614.8	14.8	2.5	982.7	82.7	9.2
3	307.2	7.2	2.4	605.0	5.0	0.8	938.7	38.7	4.3
3	298.4	1.6	0.5	649.8	49.8	8.4	967.3	67.3	7.5
3	301.5	1.5	0.5	626.2	26.2	4.4	966.0	66.0	7.3
3	298.6	1.4	0.5	650.2	50.2	8.4	956.7	56.7	6.3
4	276.3	23.7	7.9	656.3	56.3	9.4	880.2	19.8	2.2
4	322.4	22.4	7.5	654.7	54.7	9.1	910.7	10.7	1.2
4	274.3	25.7	8.6	668.1	68.1	11.4	888.1	11.9	1.3
4	288.0	12.0	4.0	661.6	61.6	10.3	897.5	2.5	0.1
5	312.8	12.8	4.3	631.7	31.7	5.3	899.8	0.2	0.0
5	296.8	3.2	1.1	641.9	41.9	7.0	965.0	65.0	7.2
5	316.2	16.2	5.4	652.5	52.5	8.8	971.1	71.1	7.9
5	312.6	12.6	4.2	664.6	64.6	10.8	962.5	62.5	7.0
Mean	300.9	14.5	4.8	640.2	40.2	6.7	943.7	48.6	5.4
SD	17.0	–	–	20.3	–	–	39.8	–	–
CV	0.06	–	–	0.03	–	–	0.04	–	–

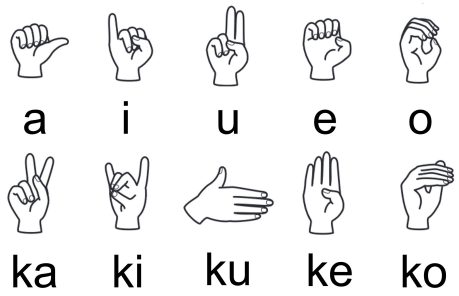


Fig. 17. Finger letters for Japanese phonetics.

Ko-e moves from "ko," with all fingers extended, to "e," with all fingers curled. The overall images of the hands are similar to one another, since the fingers are not extended upward; however, the recognition rate was not so low.

The recognition rate for a-ka was the lowest, even though the recognition rate seemed to be high, because a-ka moves from "a," with the thumb extended to "ka," with the index and middle fingers extended upward.

Table V shows recognition rate and average number of attempts per seat. The recognition rates for Seats A and C, in which the participants' right hands were photographed from behind, were slightly lower, but all recognition rates exceeded 80%, and the average number of trials was less than 2.27 times, regardless of the direction of the hand gesture.

Table VI shows recognition rate and average number of attempts per participant. Although there were differences in the recognition rates among participants, all achieved at least 80%. Participants looked at pictures of the finger letters and imitated them, but there were differences in their accuracy, which is thought to have led to the differences in their recognition rates.

TABLE IV
RECOGNITION RATE AND AVERAGE NUMBER OF ATTEMPTS PER FINGER GESTURE.

Finger letter	Recognition rate (%)	Average number of attempts
a-ka	72.7	1.84
i-ku	97.7	1.58
u-ki	93.2	2.20
o-ke	100.0	1.59
ko-e	88.6	2.74

TABLE V
RECOGNITION RATE AND AVERAGE NUMBER OF ATTEMPTS PER SEAT.

Seat	Recognition rate (%)	Average number of attempts
A	89.1	2.20
B	94.5	1.71
C	80.0	2.27
D	98.2	1.80

5) Discussion: Based on the experiment's results, the finger gestures performed within the range captured by the two cameras can be generally recognized from any direction and by all participants. However, if the gesture itself includes an action that is difficult for the participant, the recognition rate declined. Therefore, it is important to set finger gestures that are easy for system users to operate.

TABLE VI
RECOGNITION RATE AND AVERAGE NUMBER OF ATTEMPTS PER PARTICIPANT.

Participant	Recognition rate (%)	Average number of attempts
P1	100.0	1.60
P2	100.0	1.45
P3	80.0	2.50
P4	95.0	1.47
P5	80.0	1.63
P6	100.0	1.90
P7	90.0	2.56
P8	80.0	2.75
P9	100.0	1.65
P10	85.0	2.29
P11	85.0	2.29

IV. CONCLUSION

The physical search system (PSS) was developed to search for lost objects in physical space. The PSS detects all objects displaced in a physical space using two cameras and a computer. Besides voice and text input, it would be useful to use hand gestures to tell the PSS what to look for. In this paper, we investigated the accuracy rate of hand gestures to indicate the size of an object. The participants wore wristbands connected by 300, 600, and 900 mm strings on their hands. They spread their hands to the full length of the wristband strings, 300, 600, and 900 mm, while they clenched their fists, and the system measured the width between them. The results showed average absolute errors of 14.5, 40.2, and 48.6 mm in the conditions of 300, 600, and 900 mm, respectively. The system could measure the widths between users' hands with little error. Then, the recognition rates of five kinds of finger gesture series were examined. The recognition rates were from 72.7–100.0%. The finger gestures could be recognized with high accuracy if the participants could easily imitate the gestures the system had learned in advance.

In the future, a series of flow: a user inputs the features of a lost object into the PSS using hand gestures, then, the PSS finds images of the object with these features in the database, and presents where it is, will be evaluated from the aspects of the system's function and interface.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 20H04470 and 22K00211.

REFERENCES

[1] S. Kajihara, M. Okazaki, C. Oshima, and K. Nakayama, "Improving a Physical Search System that Detects Even Unknown Displaced Objects Using Image Differences," EMERGING 2022, The Fourteenth International Conference on Emerging Networks and Systems Intelligence, IARIA, pp. 13-18, 2022.

[2] S. Kajihara, M. Okazaki, K. Kawabata, H. Furukawa, C. Oshima, et al., "Proposal and verification of a physical search system that does not require pre-learning data and sensors other than cameras," IPSJ Transactions on digital practices, vol. 3, no. 2, pp. 76–92, 2022. (in Japanese)

[3] K. Muraoka and T. Terada: A Motion Recognition Method by Constancy Decision, IPSJ Journal, vol. 52, no. 6, pp. 1968-1979, 2011.

[4] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, et al., "Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor," Proceedings of the 25th annual ACM symposium on user interface software and technology, pp. 167–176, 2012.

[5] Vicon Motion Systems Ltd., VICON. Available: https://www. vicon.com/ (accessed May 20, 2023)

[6] NaturalPoint Inc., OptiTrack. Available: https://optitrack. com/ (accessed May 20, 2023)

[7] Leap Motion, Inc., Leap Motion Controller. Available: https://www.ultraleap.com/ (accessed May 20, 2023)

[8] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," In arXiv preprint arXiv:1812.08008, 2018.

[9] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. Chang, and M. Grundmann, "MediaPipe Hands: On-device Real-time Hand Tracking," Fourth CVPR Workshop on Computer Vision for Augmented and Virtual Reality, rXiv:2006.10214, 2020.

[10] Google, "MediaPipe." Available: https://google.github.io/mediapipe/ (accessed May 20, 2023)

[11] R. Hamasaki and K. Nakayama, "A deep learning system that learns a discriminative model autonomously using difference images," Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, pp. 1683–1685, 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.

[13] ResNet50. Available: https://jp.mathworks.com/help/deeplearning/ref/resnet50.html;jsessionid=c2d1fcfb1eb58ff18ab9a8beff0c (accessed May 20, 2023)

[14] ImageNet. Available: https://www.image-net.org/ (accessed May 20, 2023)

[15] B. Li and D. Lima, "Facial expression recognition via ResNet-50," International Journal of Cognitive Computing in Engineering, vol. 2, pp. 57-64, 2021.

[16] S. Bhattacharyya, "Understand and Implement ResNet-50 with TensorFlow 2.0," Towards Data Science. Available: https://towardsdatascience.com/understand-and-implement-resnet-50-with-tensorflow-2-0-1190b9b52691 (accessed May 20, 2023)

[17] C. Chen, W. Zhu, J. Steibel, J. Siegford, J. Han, et al., "Classification of drinking and drinker-playing in pigs by a video-based deep learning method," Biosystems Engineering, vol. 196, pp. 1-14, 2020.

[18] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," Proceedings of ICML 2000, vol. 1, pp. 727-734, 2000.

[19] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering," Proceedings of ICML 1998, vol. 98, pp. 91–99, 1998.

[20] imgsim. Available: https://github.com/Nr90/imgsim (accessed May 20, 2023)

[21] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," Pattern Recognition, vol. 47, no. 6, pp. 2280-2292, 2014.

[22] Logicool, "C270 web camera." Available: https://www.logitech.com/en-eu/products/webcams/c270-hd-webcam.960-001063.html (accessed May 20, 2023)

Pattern Discovery and Stylometric Analysis in English Literature and Literary Translation Through State Integration in Markovian Representations

C. H. C. Leung
School of Science and Engineering &
Guangdong Provincial Key Laboratory of Future Networks of Intelligence
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
clementleung@cuhk.edu.cn

C. J. Zeng
School of Humanities and Social Science
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
chenjiezeng@link.cuhk.edu.cn

***Abstract*—In analysing English literary work, the distinct aims and objectives are to determine the authorship, period, style, motif, and purpose. Here, the proper evaluation of results in English Literature is: first, place the known literary work in a machine learning model and discover their patterns and styles; second, compare the corresponding metrics with an unknown literary work. Since obtaining such knowledge from human experts is laborious and highly subjective, we align a data analysis method with extensions of the Markovian representations, which can be generalized to more versatile descriptions as the context develops. In particular, we consider the simple Markovian model and more elaborate generalisations that aim to remove the limitations of the memoryless properties of the basic Markovian representations. The first generalisation extends the state space by using the Cartesian product to form the composite state space, while the second approach exploits the stanza structure to integrate the states. The first approach can incorporate arbitrary long-time steps but leads to a high-dimension transition matrix. In contrast, the second more preferable approach yields a relatively small dimension matrix, which is computationally much more efficient. In addition, the latter approach also leads itself to further state integration by judiciously analysing the purpose of each line of a passage and provides the scope for analysing much larger corpora. Through the appropriate use of Markovian representation generalisations, examining the pattern of probability entries in the transition matrix, and applying this characterisation to the vast body of English literature, much more scientific, objective, and reliable decisions can be arrived at concerning proper authorship, writing style and other literary qualities.**

Keywords - Victorian novels; English poems; multi-step Markov chain; Shakespearean plays; Brontës; Sparse Matrix.

I. INTRODUCTION

This paper extends our previous paper [1] that analysed English literature by using a Markov model from a pragmatic aspect, namely, defining authorship. Still, computer-aided authorship attribution can be divided into two approaches, pragmatic and philosophical, which are related [9]. To have additional contributions, this paper uses both pragmatic and philosophical approaches to further analyse English literature and its Chinese translation by using multi-step Markov chains and expanding memory through state integration. The additional contributions of this paper comprise a new philosophical approach, a second language in literary work, and a new integration method. The analysis of vast English poems and their Chinese translation aims to determine several attributes. Such tasks may be viewed from a machine learning perspective [11] [12] [16] [17] [20], where one learns the quantitative features of known passages and uses these to determine obscure passages' attributes.

Historically, a manual approach to determining authorship primarily relied on human experts. However, it could sometimes be faulty as experts must systematically process the vast volume of literary data. For example, one poem, 'Stanzas', supposedly written by Emily Brontë, beginning 'Often rebuked, yet always back returning', is of uncertain authorship. Although Emily's sister, Charlotte Brontë, included it with the seventeen poems by Emily that she published in her selection of 'Literary Remains' that accompanied her 1850 edition of 'Wuthering Heights and 'Agnes Grey' [23].

Fortunately, unsolvable problems can be identified using machine learning. One positiveness brought by technological development is that a Machine learning approach could interplay and assist humans in a more complicated and challenging synthesis of ascertainable attributes: authorship, style, pattern, purpose, theme, and function. Stylometric analysis may indicate that a particular author's works are structurally distinct.

By using stochastic analysis, more efficient and reliable decisions can be made. Here, we shall use the Markov model's most straightforward form of dependency. The advantage of the Markov model is that introducing some dependency allows the styles to be much more richly represented than the simple independent model.

The rest of the paper is organised as follows. Section II provides a literature review, limitations, and achievements. Then using English literature examples, Section III motivates studying literary work through Markov chains. In Section IV, we explain the expansion of memory through state integration. Experiments on English passages and their Chinese translations are carried out in Section V. Finally, the paper concludes in Section VI.

II. RELATED WORKS

Previously, scholars in many fields have argued and studied the authorship issue. The historical record shows that William Shakespeare of Stratford-upon-Avon was identified as the person who was a player, a Globe shareholder, and the author of the plays and poems that carried his name. No evidence indicates

that anyone from the Elizabethan and Jacobean periods doubted this attribution [35].

On the other hand, several studies focused on authorship analysis of English literature using Markov chain models. The following studies demonstrate the effectiveness of Markov chain models in authorship analysis of English literature and suggest that more complex models, such as multi-step Markov chains or neural networks, may lead to even better performance. Specifically, Koppel et al. [36] used statistical markers, complex multivariate methods, and machine learning-based classification methods to examine the authorship. The analysis shows that two of the most sophisticated machine learning methods, SVM and Bayesian regression, offer an effective and efficient solution to the problem of authorship attribution. Also, Malyutov [37] looked at the near-optimal method based on Kolmogorov conditional complexity, attributing the discovered works of Shakespeare and those allegedly written by M. Twain, as well as binary discrimination of the Federalist papers by using Naive Bayes and other classifiers. Segarra, Eisen, and Ribeiro [38] introduced a method of authorship attribution called function word adjacency networks (WANs), which uses function words as nodes and directional edges to represent the likelihood of finding one function word in proximity to another. These WANs can be interpreted as transition probabilities of a Markov chain and are compared using relative entropies. Since function words are independent of content, their use tends to be specific to an author, making a good summary of stylometric fingerprints. Our previous papers [1] [2] used iambic pentameter to characterize these dimensions using Markov chains. We adopt a machine learning approach, processing and extracting known passages to create a signature transition matrix. Then we use a multi-step Markov chain to characterize the evolution of stress levels over time. The model can incorporate an amount of previous stress level memory, making it a flexible approach.

While Markov chains have been effective in authorship analysis of English literature, several limitations and challenges are associated with using Markov models. First, Markov chain models assume that the underlying data follows a stationary probability distribution, meaning that the statistical properties of the data do not change over time. However, in real-world scenarios, authors' writing styles may change over time for various reasons, such as personal experiences, age, or exposure to literature. Second, Markov chain models rely on selecting appropriate features and parameters to accurately represent an author's writing style. Choosing the right features and parameters can be challenging and time-consuming, requiring domain expertise and experimentation. Moreover, most of the previous studies focused on monolingual literature work in English. This paper not only analyses the monolingual literature, but also compares bilingual texts and the Chinese translation of English literature using a multi-step Markov model. Lastly, Markov chain models may struggle to differentiate between multiple authors who have similar writing styles or who have collaborated on a piece of writing. Additional techniques, such as stylometry or deep learning algorithms, may need to be employed in such cases. Therefore, while Markov chain models have been helpful in authorship analysis of English literature, they are not without their limitations and further research is required to overcome these challenges.

We mainly went through the following steps to achieve authorship analysis of English literature using multi-step Markov chain models. First, we collected data collection and gather a corpus of written works from multiple authors, such as William Shakespeare, the Brontë sisters, and W.B. Yeats. Second, we clean and preprocess the data to remove unwanted elements such as punctuation, numbers, or the last words. Then we select the representative data and extract its feature that can be used to represent the writing style of the author. For example, one or two authors could use the same rhythming patterns as features. Third, we train the model using multi-step Markov to identify each author's writing style. Fourth, to test the Markov model on a validation dataset, we evaluate its ability to correctly identify unknown texts' authorship. Therefore, we conduct two lingual experiments with the Markov chain to optimise the performance of the model in both monolingual and bilingual literature. Finally, we can use the trained model to attribute authorship of unknown texts with similar features to the data and texts we test and analyse in this paper.

III. BASIC MARKOV REPRESENTATION

Here, we illustrate the use of our approach by making use of some well-known poems, which are typical of similar poems. Charlotte Brontë has written more than 200 extant English poems, which remain of great interest as evidence of her developing ability to express emotion, her fascination with exotic characters and scenery, and her absorption of the techniques, images, and vocabulary of the poets whose work excited her. The poets that inspired Charlotte Brontë include but are not limited to William Wordsworth, William Shakespeare, and Samuel Taylor. Thus, Charlotte Brontë's poems almost always have a rich verbal texture, but her control of their style and structure is often insecure, except when she fair-copies or revises them [23]. Fortunately, among the five poems that were written or edited in 1847, two were used in 'Jane Eyre' [26]: 'My feet they are sore' (p. 22) and 'The truest love' (pp. 265-266).

English literature and poems could be differentiated from poetic devices, rhyme patterns, themes, and motifs. Still, the works written by the Brontë sisters share a lot of common grounds. For instance, we can see a common theme at the heart of all of Emily Brontë's poetry: the desire for a unified sense of the self and a simultaneous awareness and fear of the self's diffusion and fragmentation [24]. Likewise, 'Jane Eyre', written by Charlotte Brontë, is a story of enclosure and escape of 'self' and a story of the movement for freedom within the economic-social and cultural context in the Victorian era [23] [25].

Such a literary theme is rather hard to capture through a machine-learning approach. For example, in the first chapter [26], little Jane is scolded by Mrs Reed, who labels Jane's actions as 'cavillers or questioners' and demands Jane sit and remain silent until Jane can 'speak pleasantly'. To quietly perform the rebellious act and not be submissive, upset as little Jane is, she buries herself in the book entitled Bewick's History of British Birds. In the introductory pages of Bewick's book, little Jane finds comfort through pictures that portray the sea fowl inhabited by "the solitary rocks and promontories" of the Norway coast.

However, it will become attainable if we break down human experts' complicated and intuitive judgement into small elements that comprise an overall appreciation of a poem and

literature to improve. Among all the attributes, iambic pentameter, rhythm pattern, and phonetic stresses could be computer-friendly and play a key role in analysing poetry using a machine-learning approach. Iambic pentameter consists of ten syllables per line, stressing every second syllable. This creates a rhythmic pattern of one unstressed syllable followed by one stressed syllable in each pair. The rhythm pattern of iambic pentameter can be compared to the sound of a heartbeat, with each stressed syllable acting as a 'beat' in the poem. Phonetic stresses emphasise a particular syllable or word, typically due to tone, pitch, or volume. In iambic pentameter, the stress is determined by the natural accentuation and pronunciation of the words.

'Where the Northern Ocean', the first poem in 'Jane Eyre', occurs here. This poem's imaginative world of literature allows Jane's mind to escape confinement [27].

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Whe**re** the **Nort**hern **Ocean**, in **va**st **whi**rls,*

A D A D A D A C A D
^ | ^ | ^ | ^ | ^ |
*Boils **round** the **nak**ed, **mel**an**cho**ly **isles***

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Of **fart**hest **Thu**le; **and** the **Atlan**tic **surge***

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Po**urs** in a**mong** the **stor**my **Heb**ri**des**.*

a. 'Where the Northern Ocean' in 'Jane Eyre', written by Charlotte Brontë. [26]

Stressed syllables vary in strength, while unstressed syllables vary in weakness [4]. In this paper, we notate the stressed sound with a "|" marking ictic syllables and a "^" marking unstressed syllables. In this notation, a standard line of iambic pentameter would look like "^ | ^ | ^ | ^ | ^ |", where each line of verse is made up of five two-syllable iambs for a total of ten syllables. As the meter is mainly about sound, not spelling, scansion adds numbers to indicate various stress levels to realize beats and offbeats (A = lightest stress, D = heaviest stress). Scansion is the analysis of regular patterns of accented, unaccented syllables.

The arrangement of words and phrases in poetic lines reflects our custom of speaking, and of hearing each other speak, in a succession of rhythmic units; if the lines are metrical, if they make patterns out of series of lightly or firmly stressed syllables, they reflect the fact that when we speak – we speak stressed syllables with greater and lesser degrees of stress [4]. It is possible to represent the stress structure by means of a Markov chain and so provide a characterization of the literary work. The Markov matrix of the above situation, when used as a sample, can be constructed as where the four states indicate the stress levels of A, B, C, and D, respectively,

$$\begin{pmatrix} 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Although the English syllables we speak can be spoken with many degrees or shades of emphasis on loudness, sharpness, duration, and other ways of signalling importance, it seems likely that in most English speech, we perceive mainly two significant levels of stress, and that we hear a continuous series of relatively stressed and relatively unstressed syllables [4]. Similarly, the following

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*His **spark**ling **eye**s, re**pleat** with **wrath**full **fire**.*

b. The Sonnets by William Shakespeare. [22]

can be constructed simply as

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Another form of counter-lauréate authorship emerges in a more prominent place, Theseus's speech in the mid-1590s romantic comedy A Midsummer Night's Dream. Noticeably, Shakespeare's self-reflexive revision, such as inserting discourse about the 'poet' as a company for 'lunatic' and 'lover', turns a speech about the madness of love into one about the poet's role in forming an eternising state of consciousness [7].

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
More **stran**ge **than** t**ue**. I **never** may be**lieve**

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
These **an**tic **fa**bles, **nor** these **fai**ry **toy**s.

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
Lo**vers** and **madmen** have **su**ch **see**thing **brains**,

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
Such **shapi**ng **fan**tasies, **tha**t a**ppre**h**end**

c. The Sonnets by William Shakespeare. [22]

Most of the poems written by the Brontës are ended with the rhythm of 'abab' pattern, such as, 'My Feet They are Sore' (p. 22) written by Charlotte Brontë in her novel 'Jane Eyre' [26]. These two poems are fair copies or revised by Charlotte Brontës [26], thus we can see regular poetic patterns in them. Specifically, in the first stanza, the last words of each line are 'weary, wild, dreary, child'. If each of the last words is given a name using alphabetic order, we name the word of the same rhythm with the same alphabetic letter. Here in this stanza, 'weary' and 'dreary' are given a random letter 'a' because they

end with the same pronunciation of '-eary', whereas 'wild' and 'child' are given another name 'b' for ending with the '-ild' sound. The same logic applies to the remaining stanza of this poem: cdcd (the second stanza); efef (the third stanza). ghgh (the fourth stanza), ijij (the fifth stanza).

Stanza 1:

*My feet they are sore, and my limbs they are w**eary**; **(a)***

*Long is the way, and the mountains are w**ild**; **(b)***

*Soon will the twilight close moonless and dr**eary** **(a)***

*Over the path of the poor orphan ch**ild**. **(b)***

d. 'My Feet They are Sore' in 'Jane Eyre', written by Charlotte Brontë. [26]

To further analyse the poem with attributes, we can now look at the stresses of each word in each line. In a line of a poem, a foot refers to either a stressed syllable or an unstressed syllable. Both syllables then form distinct pairs, as a musical measure consists of a certain number of beats. Delimitation of the spoken chain sounds can be based on auditory impressions, but describing these sounds is an entirely different process. Description can be carried out based on the articulatory act, for it is impossible to analyze the sound units in their chain [16].

Stanza 2:

A B A C A D A B A D
^ | ^ | ^ | ^ | ^ |
*Why **did** they **send** me **so** far **and** so **lonely**, **(c)***

A B A C A B A C A C
^ | ^ | ^ | ^ | ^ |
*Up **where** the **moor**s spr**ead an**d grey **rocks** are **piled**? **(d)***

A B A D A C A D
^ | ^ | ^ | ^ |
*Men **are** hard-**hearted**, and **kind** angels **only** **(c)***

A B A C A D C D
^ | ^ | ^ | ^ |
*Watch **o'er** the **steps** of a **poor** orphan **child**. **(d)***

Stanza 3:

A D A C A C A B A C
^ | ^ | ^ | ^ | ^ |
*Yet **distant** and **soft** the **night** bree**ze is** blo**wing**, **(e)***

A B A C A B A C B C
^ | ^ | ^ | ^ | ^ |
*Clouds **there** are **none**, and **clea**r st**ar**s bea**m mild**, **(f)***

A B A C A C A D
^ | ^ | ^ | ^ |
*God, **in** His **mercy**, pro**tection** is **showing, (e)***

A B A C A D C D
^ | ^ | ^ | ^ |
*Comfor**t and** **hope** to the **poor** orphan **child**. **(f)***

Stanza 4:

A D A D A D A D
^ | ^ | ^ | ^ |
*Ev'n **should** I **fall o**'er the **broken** bridge **passing**, **(g)***

A D A D A D A D
^ | ^ | ^ | ^ |
*Or **stray** in the **marshes**, by **false** lights **beguiled**, **(h)***

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Still **will** my **Fa**ther, **wi**th **promise** and **blessing**, **(g)***

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Take **to** His **bos**om **the** poor **or**phan **child**. **(h)***

Stanza 5:

A D A D A D A D
^ | ^ | ^ | ^ |
*There **is** a **thought** that for **strength** should **avail** me, **(i)***

A D A D A D A D
^ | ^ | ^ | ^ |
*Though **both** of **shelter** and **kindred** de**spoiled**; **(j)***

A D A D A D A D A D
^ | ^ | ^ | ^ | ^ |
*Hea**ven** is a **home**, and a **rest** will **not** fail **me**; **(i)***

A D A D A D A D
^ | ^ | ^ | ^ |
*God **is** a **friend** to the **poor** orphan **child**. **(j)***

e. 'My Feet They are Sore' in 'Jane Eyre', written by Charlotte Brontë. [26]

The advantage of Markov models [13] [18] for analysing sequential data is that segmentation and classification are performed simultaneously in an integrated procedure. Using efficient and robust training and decoding algorithms, Markov model recognition systems can effectively be realized for large-scale classification systems [3].

The poem reappears in a different context when Jane and Rochester declare their love for each other on Midsummer Eve [26]. It serves a new purpose: to symbolise the to-be Mrs Rochester's potentialities for freedom and happiness, as well as the natural affinity between her and Rochester. The poem that starts with 'the truest love that ever heart' is narrated from Rochester's perspective, indicating a potential marriage and freedom afterwards.

Stanza 1:

A D A D A D A D
^ | ^ | ^ | ^ |
*The **tru**est **love** that **ev**er **heart***

A D A D A D
^ | ^ | ^ |
*Fel**t at** its **kin**dled **core**,*

A D A D A D A D
^ | ^ | ^ | ^ |
Did **through** each **vein**, in **quicken**ed **start**,

A D A D A D
^ | ^ | ^ |
The **tide** of **bei**ng **pour**.

A D A D A D A D
^ | ^ | ^ | ^ |
Her **com**ing **was** my **hope** each **day**,

A D A D A D
^ | ^ | ^ |
Her **part**ing **was** my **pain**;

A D A C A D A D
^ | ^ | ^ | ^ |
The **chance** that **did** her **steps** de**lay**

A D A D A D
^ | ^ | ^ |
Was **ice** in **ev**ery **vein**.

Stanza 2:

A D A D A D A D
^ | ^ | ^ | ^ |
I **dreamed** it **would** be **name**less **bliss**,

A D A D A D
^ | ^ | ^ |
As **I** loved, **loved** to **be**;

A D A D A D
^ | ^ | ^ |
And **to** this **object** did I **press**

A D A D A D
^ | ^ | ^ |
As **blind** as **ea**ger**ly**.

Stanza 3:

A D A D A D A D
^ | ^ | ^ | ^ |
But **wide** as **path**less **was** the **space**

A D A D A D
^ | ^ | ^ |
That **lay** our **lives** bet**ween**,

A D A C A D A D
^ | ^ | ^ | ^ |
And **dange**rous **as** the **foa**my **race**

A D A D A D
^ | ^ | ^ |
Of **oce**an-**sur**ges **green**.

Stanza 4:

A D A D A D A D
^ | ^ | ^ | ^ |
And **haun**ted **as** a **rob**ber-**path**

A D A D A D
^ | ^ | ^ |
Through **wild**er**ness** or **wood**;

A D A D A D A D
^ | ^ | ^ | ^ |
For **Might** and **Right**, and **Woe** and **Wrath**,

A D A D A D
^ | ^ | ^ |
Bet**ween** our **spir**its **stood**.

Stanza 5:

A D A D A D A D
^ | ^ | ^ | ^ |
I **dan**gers **dared**; I **hind**rance **scorned**;

A D A D A D
^ | ^ | ^ |
I **om**ens **did** de**fy**:

A D A D A D A D
^ | ^ | ^ | ^ |
Wha**te**ver **men**aced, **hara**ssed, **warned**,

A D A D A D
^ | ^ | ^ |
I **pass**ed im**petu**ous **by**.

Stanza 6:

A D A D A D A D
^ | ^ | ^ | ^ |
On **sped** my **rain**bow, **fast** as **light**;

A D A D A D
^ | ^ | ^ |
I **flew** as **in** a **dream**;

A D A D A D A D
^ | ^ | ^ | ^ |
For **glor**ious **rose** up**on** my **sight**

A D A D A D
^ | ^ | ^ |
That **child** of **Shower** and **Gleam**.

Stanza 7:

A D A D A D A D
^ | ^ | ^ | ^ |
Still **bright** on **clouds** of **suffer**ing **dim**

A D A D A D
^ | ^ | ^ |
*Shines **that** soft, **sol**emn **joy**;*

A D A D A D A D
^ | ^ | ^ | ^ |
*Nor **care** I **now**, how **dense** and **grim***

A D A D A D
^ | ^ | ^ |
*Di**sas**ters **gat**her **nigh**.*

Stanza 8:

A D A D A D A D
^ | ^ | ^ | ^ |
*I **care** not **in** this **mom**ent **sweet**,*

A D A D A D A D
^ | ^ | ^ | ^ |
*Though **all** I **ha**ve **ru**shed **o'er***

A D A D A D A D
^ | ^ | ^ | ^ |
*Should **come** on **pin**ion, **strong** and **fleet**,*

A D A D A D
^ | ^ | ^ |
*Pro**claim**ing **vengea**nce **sore**:*

Stanza 9:

A D A D A D A D
^ | ^ | ^ | ^ |
*Though **haug**hty **Hate** should **strike** me **down**,*

A D A D A D
^ | ^ | ^ |
*Right, **bar** app**roach** to **me**,*

A D A D A D A D
^ | ^ | ^ | ^ |
*And **grin**ding **Might**, with **furi**ous **frown**,*

A D A D A D
^ | ^ | ^ |
*S**wea**r **end**less **enmity**.*

Stanza 10:

A D A D A D A D
^ | ^ | ^ | ^ |
*My **love** has **plac**ed **her** little **hand***

A D A D A D
^ | ^ | ^ |
*With **no**ble **faith** in **mine**,*

A D A D A D A D
^ | ^ | ^ | ^ |
*And **vowed** that **wed**lock's **sac**red **band***

A D A D A D
^ | ^ | ^ |
*Our **nat**ure **sha**ll en**twine**.*

Stanza 11:

A D A D A D A D
^ | ^ | ^ | ^ |
*My **love** has **sworn**, with **seal**ing **kiss**,*

A D A D A D
^ | ^ | ^ |
*With **me** to **live**—to **die**;*

A D A D A D A D
^ | ^ | ^ | ^ |
*I **have** at **last** my **name**less **bliss**.*

A D A D A D
^ | ^ | ^ |
*As **I** love—**loved** am **I**!*

f. 'The Truest Love' in 'Jane Eyre' by Charlotte Brontë. [26]

The above passage may be characterized by the following matrix

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Thus, the authorship may be evaluated by examining the pattern of entries in the transition matrix.

IV. Extending Memory Through State Integration

The basic Markovian representation, while useful, suffers from the limitations that plague all basic Markovian models in that the memory of any previous states is completely erased.

However, it is possible to inject limited memory by forming the Cartesian product of the basic states, but this tends to be computationally prohibitive. In our situation above, incorporating one-step memory requires augmenting the state space to 16, and incorporating two-step memory requires augmenting the state space to 64, resulting in a 64 × 64 transition matrix. Thus, incorporating k-step memory requires augmenting the state space to 4^{k+1} with a rather unwieldy $4^{k+1} \times 4^{k+1}$ transition matrix:

$$\begin{pmatrix} e_{11} & e_{12} & e_{13} & \dots & e_{1n} \\ \dots & \dots & & & \\ e_{i1} & e_{i2} & \dots & & e_{in} \\ \dots & \dots & & \dots & \dots \\ \dots & \dots & & & \\ e_{n1} & e_{n2} & \dots & & e_{nn} \end{pmatrix} \quad (1)$$

where each $e_{ij} \geq 0$, and

$$\sum_{j=1}^{n} e_{ij} \leq 1, \qquad i = 1, 2, \ldots, n.$$

Here, we allow the possibility of

$$\sum_{j=1}^{n} e_{ij} < 1,$$

which can sometimes occur, especially when absorbing states may be suitably identified [34]. Techniques for dimension reduction of the above matrix have been proposed in [1], where, for example, a 16 × 16 transition matrix may be reduced to a 6 × 6 matrix.

Instead of the above approach, however, we shall exploit the line structure in each stanza. Consider Stanza 4 of 'My Feet They Are Sore' in 'Jane Eyre', written by Charlotte Brontë above. We observe that line 1 and line 2 have the same structure, which we can represent by ξ, while line 3 and line 4 have the same structure, which we can represent by ζ. Thus, in Stanza 4, we have the transition structure

$\xi \rightarrow \xi$ (probability ⅓)
$\xi \rightarrow \zeta$ (probability ⅓)
$\zeta \rightarrow \zeta$ (probability ⅓)
$\zeta \rightarrow \xi$ (probability 0).

This gives the 2 × 2 transition matrix

$$\begin{pmatrix} ⅓ & ⅓ \\ 0 & ⅓ \end{pmatrix}$$

where the first state corresponds to ξ, and the second state corresponds to ζ.

Applying the same framework to the first stanza of Rochester's poem, we introduce a further state λ corresponding to the second line of Rochester's poem, which then yields the transition structure

$\lambda \rightarrow \xi$ (probability 3/7)
$\xi \rightarrow \xi$ (probability 0)
$\xi \rightarrow \lambda$ (probability 4/7)
$\lambda \rightarrow \lambda$ (probability 0)
$\lambda \rightarrow \zeta$ (probability 0)
$\zeta \rightarrow \lambda$ (probability 0).
$\zeta \rightarrow \xi$ (probability 0)
$\xi \rightarrow \zeta$ (probability 0)
$\zeta \rightarrow \zeta$ (probability 0)

This will give a 3 × 3 transition matrix with all diagonal elements equal to 0.

$$\begin{pmatrix} 0 & 0 & 4/7 \\ 0 & 0 & 0 \\ 3/7 & 0 & 0 \end{pmatrix}$$

While this is a more complete representation, it will be computationally less efficient and is only necessary when we need to combine the above two passages. In analyzing individual passages, using a 2 × 2 transition matrix may sometimes suffice, which on suitably defining the states, yields the following simpler transition matrix

$$\begin{pmatrix} 0 & 4/7 \\ 3/7 & 0 \end{pmatrix}$$

Following the simplified approach, the second stanza of the same poem yields the transition structure

$\zeta \rightarrow \xi$ (probability ⅔)
$\xi \rightarrow \xi$ (probability 0)
$\xi \rightarrow \zeta$ (probability ⅓)
$\zeta \rightarrow \zeta$ (probability 0).

This gives the transition matrix

$$\begin{pmatrix} 0 & ⅓ \\ ⅔ & 0 \end{pmatrix}$$

Another way to avoid dealing with large transition matrices, judgment can be exercised to integrate the states further. In this way, considering large passages will be less unwieldy. From the recital perspective, it may be possible to merge the states ζ and λ and regard them as serving the same purpose. Let us map both states to η. Then we have the states ξ and η. In so combining the two stanzas will yield the following transition structure

$\eta \rightarrow \xi$ (probability 6/11)
$\xi \rightarrow \xi$ (probability 0)
$\xi \rightarrow \eta$ (probability 5/11)
$\eta \rightarrow \eta$ (probability 0),

which yields the combined transition matrix

$$\begin{pmatrix} 0 & 5/11 \\ 6/11 & 0 \end{pmatrix} \quad (2)$$

Evidently, this matrix is computationally much more efficient than the matrix (1), while the amount of memory in the underlying Markovian model (2) is no less substantial.

V. EXPERIMENTATION

In this section, we conduct two experiments: monolingual and bilingual experiment. In a monolingual experiment, we analyse English stanzas written by Emily Brontë. Whereas in a bilingual experiment, we first analyse an English poem, 'Leda and the Swan' written by W. B. Yeats, then compare its Chinese versions translated by Mu Yang and Guangzhong Yu.

A. Monolingual Experiment

Ranked with Elizabeth Barrett Browning, Christina Rossetti, and Emily Dickinson, Emily Brontë is one of the pre-eminent women poets of the Victorian period [23]. As mentioned in the introduction of this paper, the poem, 'Stanzas' beginning 'Often rebuked, yet always back returning', is of uncertain authorship. Supposedly, it was written by Emily Brontë. Charlotte substantially revised and retitled most of these poems, and editors now print Emily's version from her manuscript (except in the case of 'Often rebuked'). Charlotte clarified in her prefatory note to 'Selections from the Poems by Ellis Bell' that 'it would not have been difficult to compile a volume out of the papers left by my sisters' [23].

Stanza 1:

*Often rebuked, yet always back retur**ning** (a)*

*To those first feelings that were born with **me**, (b)*

*And leaving busy chase of wealth and lear**ning** (a)*

*For idle dreams of things which cannot **be**: (b)*

Stanza 2:

*To-day, I will seek not the shadowy re**gion**; (c)*

*Its unsustaining vastness waxes dr**ear**; (d)*

*And visions rising, legion after le**gion**, (c)*

*Bring the unreal world too strangely n**ear** (d)*

Stanza 3:

*I'll walk, but not in old heroic tr**aces**, (e)*

*And not in paths of high morali**ty**, (f)*

*And not among the half-distinguished f**ace**s, (e)*

*The clouded forms of long-past histo**ry**. (f)*

Stanza 4:

*I'll walk where my own nature would be lea**ding**: (g)*

*It vexes me to choose another gu**ide**: (h)*

*Where the grey flocks in ferny glens are feed**ing**; (g)*

*Where the wild wind blows on the mountain s**ide**. (h)*

Stanza 5:

*What have those lonely mountains worth revea**ling**? (i)*

*More glory and more grief than I can t**ell**: (j)*

*The earth that wakes one human heart to feel**ing** (i)*

*Can centre both the worlds of Heaven and H**ell**. (j)*

g. 'Stanzas' begins with 'Often rebuked' by Emily Brontë, edited by Charlotte Brontë. [28]

Then, before a further elaboration of the remaining poem, Charlotte writes, 'The following are the last lines my sister Emily ever wrote.' [28]

No coward soul is mine,

No trembler in the world's storm-troubled sphere:

I see Heaven's glories shine,

And faith shines equal, arming me from fear.

O God within my breast,

Almighty, ever-present Deity!

Life—that in me has rest,

As I—undying Life—have power in thee!

Vain are the thousand creeds

That move men's hearts: unutterably vain;

Worthless as withered weeds,

Or idlest froth amid the boundless main,

To waken doubt in one

Holding so fast by thine infinity;

So surely anchored on

The stedfast rock of immortality.

With wide-embracing love

Thy spirit animates eternal years,

Pervades and broods above,

Changes, sustains, dissolves, creates, and rears.

Though earth and man were gone,

And suns and universes ceased to be,

And Thou were left alone,

Every existence would exist in Thee.

There is not room for Death,

Nor atom that his might could render void:

Thou—THOU art Being and Breath,

And what THOU art may never be destroyed.

h. 'Stanzas' begins with 'Often rebuked' by Emily Brontë, edited by Charlotte Brontë. [28]

The above poem is claimed to be written by Emily Brontë and edited by Charlotte Brontë. Nevertheless, the authorship is uncertain [26]. We analyse this piece of work as the dataset for comparison and evaluation. There is a possibility that the writers' writing style varies from time to time. Therefore, the loose verification of late Brontës may be significantly different from their early works [9].

Given a close look at the poem 'Stanzas', we can spot a similar pattern and rhythm in it. For example, the number of lines and the rhythm pattern in the above 'Stanzas' that begins with 'Often rebuked' written by Emily Brontë, edited by Charlotte Brontë is akin to the poem 'My Feet They are Sore' in 'Jane Eyre' written by Charlotte Brontë. They end with the 'abab, cdcd, efef, ghgh, ijij' pattern. More specifically, in the first stanza of the poem 'Stanzas', the rhythm of the first and third lines ends with '-ning', which is labelled by a random 'a', and the rhythm of the second and fourth lines end with '-e', which is marked as a sequential letter 'b'. The following rhythm in the 'Stanzas' edited by Charlotte Brontë is aligned with the pattern in her poems included in 'Jane Eyre'. Such identical authorship features can be easily found through Markov Model.

B. Bilingual Experiment

W. B. Yeats's poem 'Leda and the Swan' is based on a Greek story in which the god Zeus swooped down and hit Leda in the form of a swan, a human and ancient Greek queen. Consequently, such misconduct led to the Trojan War. Seemingly ironic, the poem could allude to the colonial relationship between Great Britain and Ireland, more specifically, to the Irish War for Independence.

Stanza 1:

Line 1: A sudden blow: the great wings beating still (k)

Line 2: Above the staggering girl, her thighs caressed (l)

Line 3: By the dark webs, her nape caught in his bill, (k)

Line 4: He holds her helpless breast upon his breast. (l)

Stanza 2:

Line 5: How can those terrified vague fingers push (m)

Line 6: The feathered glory from her loosening thighs? (n)

Line 7: And how can body, laid in that white rush, (m)

Line 8: But feel the strange heart beating where it lies? (n)

Stanza 3:

Line 9: A shudder in the loins engenders there (o)

Line 10: The broken wall, the burning roof and tower (p)

Line 11: And Agamemnon dead.

Line 12: Being so caught up, (q)

Line 13: So mastered by the brute blood of the air, (o)

Line 14: Did she put on his knowledge with his power (p)

Line 15: Before the indifferent beak could let her drop? (q)

i. 'Leda and the Swan' by W. B. Yeats. [31]

Analysing Yeats's poem through a human's close reading from a literature perspective, the poem consists of three concepts: defamiliarisation, paradox, and textual indeterminacies. The world would become strange due to the act of defamiliarisation. In the first stanza, 'sudden, still, staggering' suggests that the story takes place in *medias res*, defamiliarising the familiar. Through the angle of the third party, Yeats creates an impression to the readers who are not notified of anything that suddenly, everything is happening from nowhere. Because of the unawareness, audiences like the blind, who can only touch a part of an elephant at once, feel unfamiliar with this pornographic scene. Such a defamiliarised act 'presents objects or experiences from an unusual perspective or in unconventional and self-conscious language that our habitual, ordinary, rote perceptions of those things are disturbed [30]'. The estranging lines in the last stanza also indicate defamiliarisation that the poetic form and language are intentionally made strange after the 'Agamemnon dead', indicating the tragedy of such an irresponsible act of sexuality.

Power imbalance, *id est* who overpowers whom, remains an unsolved and ambiguous question for readers. The textual indeterminacy could lie in 'Leda's profound and provocative dramatisation of the ambiguities of sexual encounters' [29]. Conventionally speaking, 'helpless, terrified' could possibly imply Leda's despair because Zeus, in the form of a swan, rapes Leda. However, without pinpointing commas, it is not absolutely clear who is helpless. We may read that ambiguity as the basis of assuming interchangeable roles, both Leda and the swan being potentially the rapist [29]. Another indeterminacy example in stanza two leads to a question: whose 'body' feels whose 'strange heart beating', which resulting audiences' imagination of what suits their habitus.

The by-product of textual indeterminacy is the paradox regarding the definition of and separation between rightness and wrongness in Yeats' poem. The sexual intercourse between the helpless and the powerful may suggest the intertwining essence of the paradox between rightness and wrongness. The presumably mighty 'glory' and helpless Leda would never foresee that their sexual affairs would 'engender' the 'broken wall, burning roof' causing 'dead, blood'. By asking three questions (two questions begin with 'how' in lines 5-8, and one begins with 'did' in lines 14-15), Yeats implies that the natural bond and sexual desire from both parties are intertwined and inseparable, and so do the paradoxical rightness and wrongness in Zeus and Leda. As a human, Leda sees Zeus as an animal; thus, sex with such a 'feathered glory' is prohibited. She intends to 'push' him from 'her loosening thighs'. However, her feeling and, more importantly, Zeus's power takes over Leda, who is therefore subordinated by power, emotion, and sexuality.

From a machine learning perspective, the above-mentioned three key concepts can be captured and simplified by analysing the form of the original and translated texts. Here we provide two different versions in Chinese for the bilingual experiment.

The original English version of 'Leda and the Swan' (also called source text) has distinct machine-learning-friendly features, especially rhythming. For example, in stanza one, the ending rhythm for lines one and three is '-ill', which we name the 'k' line; in contrast, the second and fourth lines end with '-ssed/st', which we call it the 'l' line. We can see the pattern regularly goes in the 'klkl' pattern. A similar pattern and feature can be seen in the second and third stanzas, which end with the rhythm of 'mnmn' and 'opqopq', respectively. The random letter 'm' represents the rhythm of '-ush' in lines five and seven; 'n' refers to the identical ending rhythm of '-ighs/ies'.

Using the Markov model, if such a feature can be primed, detected, and memorised, then this model can be used to predict the translated text (also known as the target text). In other words, such rhythming features in the English source text should ideally be translated by a faithful translator to achieve the same effect in Chinese-translated poems.

遽然的垂擊：巨翼猶拍打於
暈眩無力的女子之上，她的雙股
被黑色的腳蹼撫弄，頸為喙所擒，
他把她無依的胸乳緊納入懷。

那些恐慌猶疑的手指怎麼可能
將插翼的光輝自漸漸鬆弛的股間推開？
而身體，在那白色的疾撞之下，
如何不覺察一奇異的心在那裡跳動？

腰際一陣戰慄於焉產生
是毀頹的城墻，塔樓熾烈焚燒
而阿加梅儂死矣。
被如此攫獲著，
如此被蒼天一狂猛的血力所制服，
她可曾利用他的威勢奪取他的洞識
在那冷漠的鳥喙廢然鬆懈之前？

j. 'Leda and the Swan' translated by Mu Yang. [32]

However, in the first piece of Chinese translation written by Yang [32], it is unlikely to see such features of rhythming if we look at the last words of each line. For example, in the first stanza, we have to end words pronounced respectively 'yu, gu, qin, huai', which do not comply with the source text's rules and writing convention, largely distinct from Yeats's writing habits. In the second stanza, the ending pronunciations are 'neng, kai, xia, dong', which are significantly different from the rhythm format in the source text. As the different rhythms cannot be memorized by a simple step of the Markov chain, the undetectability of the rhythm in Yang's translation could fail and an undesired translation from a machine learning perspective.

Stanza 1:

Line 1:猝然一攫：巨翼猶兀自拍動，(k')
Line 2:扇著欲墜的少女，他用黑蹼(l')
Line 3:摩挲她雙股，含她的後頸在喙中，(k')
Line 4:且擁她捂住的乳房在他的胸脯。(l')

Stanza 2:

Line 5:驚駭而含糊的手指怎能推拒，(m')
Line 6:她鬆弛的股間，那羽化的寵倖？(n')
Line 7:白熱的衝刺下，那撲倒的凡軀(m')
Line 8:怎能不感到那跳動的神異的心？(n')

Stanza 3:

Line 9:腰際一陣顫抖，從此便種下(o')
Line 10:敗壁頹垣，屋頂和城樓焚毀，(p')
Line 11:而亞加曼儂死去。
Line 12: 就這樣被抓，(q')
Line 13:被自天而降的暴力所凌駕，(o')
Line 14:她可曾就神力汲神的智慧，(p')
Line 15:乘那冷漠之喙尚未將她放下？(q')

k. 'Leda and the Swan' translated by Guangzhong Yu. [33]

On the contrary, Yu's version of translation [33] aligns with Yeats's rhythming format. Such rhythming features in Yu's translation could be memorized by the Markov model. For instance, in stanza one of Yu's translation, lines one and three end with the rhythm '-ong', and the rhythm of lines two and four are identical 'pu'. Likewise, the rhythming pattern in stanza two goes 'u' (lines five and seven) and 'xin' (lines six and eight), which suggests that Yu has captured the structure feature and beauty in Yeats's poem and then strategically and equivalently translated them into Chinese.

VI. CONCLUSION

In carrying out the analysis of the works in English and other literature, there are definite requirements, which include determining the authorship, dating the period of the work, and establishing the passage's style, theme, and purpose. We have placed such tasks in a machine-learning context, where a learning phase involving known passages is followed by a testing phase involving unknown passages.

Since obtaining such knowledge from human experts is time-consuming, subjective, and error-prone, we combine a data analysis approach with Markov models. The Markov model can represent and encapsulate the sequential flow of writing characteristics in passages. In addition, we also proposed two strategies to overcome the memoryless properties of the Markov model. The first involves augmenting the state space of the Markovian representation by repeatedly forming the Cartesian product of the underlying space. However, while this is a mathematically versatile approach, it will lead to high computational costs. The second approach exploits a poem's stanza and line structure, which has shown to be much more efficient, yielding a much smaller dimension transition matrix. In addition, the latter approach also lends itself to further state integration by judiciously analysing the purpose of each line of a passage and providing the scope for analysing much larger corpora. Through the appropriate use of Markovian variants, examining the pattern of probability entries in the transition matrix, and applying this characterisation to the vast body of English literature, more scientific, objective, and reliable decisions can be made concerning proper authorship, writing style, and other literary qualities.

REFERENCES

[1] C. H. C. Leung and C. Zeng. The Use of Multi-Step Markov Chains in the Characterization of English Literary Works. In Procedings of the 11th International Conference on Data Analytics, Valencia, Spain, pp. 43-48, 2022.

[2] C. Zeng and C. Leung, The Use of Stochastic Models in the Analysis of Vast English Literary Data Corpora, 2020 6th International Conference on Big Data and Information Analytics (BigDIA), Shenzhen, China, pp. 282-288, 2020.

[3] T. Plotz and G. A Fink, Markov Models for Handwriting Recognition. London: Springer London, 2011.

[4] G. T. Wright, *Shakespeare's Metrical Art.* Berkeley: University of California Press, 1988.

[5] W. Shakespeare and P. Alexander, *William Shakespeare; the complete works*. London: Collins, 1964.

[6] G. Taylor, Shakespeare and Others: The Authorship of "Henry the Sixth, Part One". *Medieval & Renaissance Drama in England*, Vol. 7 pp. 145-205. Rosemont Publishing & Printing Corp DBA Associated University Presses, 1995.

[7] C. Patrick, *Shakespeare's literary authorship*. Cambridge: Cambridge University Press, 2008.

[8] P. Edmondson and S. Wells, *Shakespeare Beyond Doubt*. Cambridge: Cambridge University Press, 2013.

[9] G. Taylor and G. Egan, *The New Oxford Shakespeare: Authorship Companion*. Oxford: Oxford University Press, 2017.

[10] E. Martina, The Use of Dialects and Foreign Languages in Shakespeare's King Henry V—Characteristics of the Fool Explored. *English Studies*, vol. 100, pp. 767-784. Colchester, Informa UK Limited, June 2019.

[11] D. Berend and A. Kontorovich, A Finite Sample Analysis of the Naive Bayes Classifier. *Journal of Machine Learning Research*, 16(1), pp. 1519-1545, 2015.

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

[13] W. Feller, *Introduction to Probability Theory and Its Applications*, Volume I, 3rd. Ed. Wiley, 2008.

[14] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[15] D. D. Lewis and W. A. Gale, A Sequential Algorithm for Training Text Classifiers. In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3-12, 1994.

[16] N. L. J. Kuang and C. H. C. Leung, Analysis of Evolutionary Behavior in Self-Learning Media Search Engines, in Proceedings of the IEEE International Conference on Big Data, Los Angeles, USA, pp. 643-650, 2019.

[17] N. L. J. Kuang and C. H. C. Leung, Performance Dynamics and Termination Errors in Reinforcement Learning - A Unifying Perspective. In Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering, pp. 129-133, 2018.

[18] E. Parzen. *Stochastic Processes*. Dover, 2018.

[19] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, Cheap and Fast - but is It Good? Evaluating Non-expert Annotations for Natural Language Tasks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 254-263, 2008.

[20] N. L. J. Kuang and C. H. C. Leung, Leveraging Reinforcement Learning Techniques for Effective Policy Adoption and Validation. in Misra S. et al. (eds) in Computational Science and Its Applications – ICCSA 2019, 311-322, Lecture Notes in Computer Science, Vol. 11620. Springer, 2019.

[21] N. L. J. Kuang and C. H. C. Leung, Performance Effectiveness of Multimedia Information Search Using the Epsilon-Greedy Algorithm, in Proceedings of the IEEE International Conference on Machine Learning and Applications, Florida, USA, pp. 929-936, 2019.

[22] W. Shakespeare, *The Sonnets.* London: Macmillan Collector's Library, 2016.

[23] C. Alexander and M. Smith *The Oxford Companion to the Brontës*. Oxford University Press, 2006.

[24] L. Pykett, *Emily Brontë*. Rowman & Littlefield Publishers, 1989.

[25] S. M. Gilbert and S. Gubar, *The madwoman in the attic: the woman writer and the nineteenth-century literary imagination.* Yale University Pres, 1979.

[26] C. Brontë, *Jane Eyre*, Oxford University Press, 2019.

[27] L. Judith and P. Christopher, From the Red Room to Rochester's Haircut: Mind Control in 'Jane Eyre'. *ESC: English Studies in Canada, 32*(4), pp. 169-188, 2008.

[28] A. Brontë, C. Brontë, and E. Brontë, *Poems by Currer, Ellis, and Acton Bell.* Project Gutenberg, 1997.

[29] WC. Barnwell, The Rapist in "Leda and the Swan". *South Atlantic Bulletin*, 42.1, pp. 62-68, 1977.

[30] J. Rivkin and R. Michael, *Literary Theory.* Blackwell Anthologies. 3rd ed. New York:Wiley, 2017.

[31] W. B. Yeats, Leda and the Swan. *Modern English Literature*, p.8, 1935

[32] M. Yang (楊牧), Translated poem of 'Leda and the Swan' 譯作《麗達與天鵝》. *Yi Shi (譯事)*. Cosmos Books Ltd. (天地圖書有限公司), p. 26, 2007.

[33] G. Yu (余光中), Translated poem of 'Leda and the Swan' 譯作《麗妲與天鵝》. *Songs of Innocence: An Anthology of Guangzhong Yu's Translated Poems* (天真的歌 : 余光中經典翻譯詩集). Jiangsu Phoenix Literature and art publishing, Ltd. (江蘇鳳凰文藝出版社), p. 76, 2019.

[34] R. A. Howard, Dynamic Probabilistic Systems, Volume I, Dover, 2017.

[35] T. Reedy and D. Kathman, *How We Know That Shakespeare Wrote Shakespeare: The Historical Facts*. Kathman & Ross, Shakespeare Authorship Page, 2005.

[36] M. Koppel, J. Schler, and S. Argamon, Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology*, 54(4), pp. 9-26, 2009.

[37] M.B. Malyutov, Authorship attribution of texts: A review. *General Theory of Information Transfer and Combinatorics*, pp.362-380, 2006.

[38] S. Segarra, M. Eisen, and A. Ribeiro, Authorship attribution through function word adjacency networks. *IEEE Transactions on Signal Processing*, *63*(20), pp. 5464-5478, 2015.

A Graph Matching Algorithm to Extend Software Wise Systems with Human Semantic

Abdelhafid Dahhani
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: abdelhafid.dahhani@univ-smb.fr
0000-0001-6314-662X

Ilham Alloui
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: ilham.alloui@univ-smb.fr
0000-0002-3713-0592

Sébastien Monnet
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: sebastien.monnet@univ-smb.fr
0000-0002-6036-3060

Flavien Vernier
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: flavien.vernier@univ-smb.fr
0000-0001-7684-6502

***Abstract*—Wise systems refer to distributed communicating software objects, which we termed Wise Objects, able to autonomously learn how they are expected to behave and how they are used. Wise Objects are designed to be associated either with software or physical objects (e.g., home automation) to adapt to end users while demanding little attention from them. This last requirement obeys to the principle of calm technology introduced by Mark Weiser and John Seely Brown in 1995. Wise Objects are endowed with autonomous computing capabilities as they implement the notion of IBM's 4 state loop Monitor-Analyze-Plan-Execute over a shared Knowledge. However, they suffer from a lack of semantic, which prevents them from communicating effectively with a human. The work presented in this paper aims at extending Wise Objects with the ability to use human semantic to communicate with a user. Construction of such systems requires at least two views: (i) a conceptual view relying on knowledge given by developers to either control or specify the expected system behavior; and (ii) an auto-generated view acquired by wise systems during their learning process. The problem is that, while a conceptual view is understandable by humans (i.e., developers, experts, etc.), a view generated by a software system contains mainly numerical information with mostly no meaning for humans. In this paper, we address the issue of how to relate both views using two state-based formalisms: Input Output Symbolic Transition Systems for conceptual views and State Transition Graphs for views generated by the wise systems. Our proposal is to extend the generated knowledge with the conceptual knowledge using a matching algorithm founded on graph morphism. Target results are twofold: (i) make wise systems' generated knowledge understandable by humans, (ii) enable human evaluation of wise systems' outputs. To illustrate the overall process, the construction of two samples of graph matching on a roller shutter and a light bulb are considered.**

***Keywords*—statecharts; monitoring systems; adaptive system and control; knowledge-based systems; discrete-event systems; graph matching; semantic.**

I. Introduction

In recent years, software technology has exponentially undergone a huge evolution increasing the development of intelligent applications using AI models and techniques, in particular machine learning techniques. Examples of AI-based systems are home-automation and software network traffic analysis. Artificial intelligence (AI) and software engineering (SE) are old interdependent and mutually beneficial fields that came into the spotlight with the advent of deep learning. The challenges for SE are linked to modeling, implementing and testing software and systems that integrate AI. When AI provides software with ability to adapt to their environment, designing such software requires dedicated approaches and tools to manage this ability leading to a non predictable software behavior. Hence, the idea underlying the Wise Systems (WS) [1] is to provide developers with software support, to help them design and build intelligent systems and applications. Indeed the availability of machine learning libraries and off-the-shelf solutions sometimes gives the illusion that developing AI-based software applications is easy, but the reality is that developing viable and trusted AI systems requires significant effort [2]–[5] and combination of AI and SE.

Wise systems refer to distributed communicating software objects, which we termed Wise Objects (WOs), able to autonomously learn how they are expected to behave and how they are used. A desirable feature of a WS is self-adaptation: the WS should be able to autonomously adapt according to their use. It can be seen as a particular Multi-Agent System [6][7] that monitors only its internal changes and does not directly observe its external environment. Concretely, a WO is a piece of software able to monitor itself: the way it is used and the way it could be used (through introspection). The main specificity of a WO is that it is able to learn about itself in an autonomous way: it monitors its method invocations and their impact, it can also simulate method invocations to envision possible use and explore/discover new states. A method implements a service or functionality to be provided by a WO Then, the collected monitoring data can feed a learning process to be able to determine usual and unusual

behavior (for instance), this learning process is implemented by experts through plugins [8]. Let us note that a plugin is software that adds new abilities or extends existing ones. We have developed several plugins for WOs, e.g., Analyzers, Planners, Graph Matchers, AI Models, etc. An example is a home-automation system that collects someone's behavior within a place and analyzes it to be able to act "silently" when necessary. Such systems should require the minimum attention from their end users while being able to adapt to changes in their behavior.

To meet those requirements and as the development of such systems is non-trivial, we developed an object based framework named Wise Object Framework [9] (WOF) to help developers design, deploy and evolve WSs. Generally, knowledge in AI-enabled systems can be provided according to two ways: describing a priori the arrangement of activities to be performed by the system, or, letting the system acquire the required knowledge using learning mechanisms.

In the former case, ontologies and/or scenarios are usually used to describe the arrangement of activities to achieve a goal as in [10][11]. In [10], functional behavior as well as interoperation of system entities are described a priori using state-diagrams. Reference [11] goes a step forward by combining ontologies to design ambient assisted living systems with specifications based on logic and analyzers to check in logic clauses before system deployment to create relevant scenarios. In those approaches, the end user is at the heart of the scenario creation process, as described in [12][13]. *In the second case*, knowledge is provided by the AI-enabled system in representations and views not necessarily understandable by humans. This relates to the wide problem of comprehensibility of AI and to the distance between the business domain and technological domain views [14].

In this context, the WO acquires by itself knowledge about its capabilities – services to be provided – and its use to moderate attention from the end-users [15] (Calm technology [16]). Mark Weiser and John Seely Brown [17] describe calm technology as "that which informs but does not demand our(users) focus or attention". The WO also analyzes this knowledge to generate new one. As a result, it produces a State Transition Graph (STG) of the WO behavior. We consider STGs as the most natural way to model system dynamics. More precisely, this graph is built by iteration, i.e., step-wise construction, during a process called introspection. This process is launched during a phase called dream phase in which the WO discovers all its states (configurations) [18]. The downside of an STG generated by a WO is that numeric data provided has no meaning for humans. In the literature [19][20], other graphs like Input Output Symbolic Transition Systems (IOSTSs) are often used by developers/experts to model the behavior of systems to manage them using oracle or controller synthesis. Since this type of graph enables conceptual models understandable by humans, it can increase the knowledge of WOs and bring semantic to STGs.

This paper extends [1], which consists in enhancing the generated knowledge with the conceptual knowledge using a matching algorithm based on graph morphism [21][22]. This provides the ability to make WSs' generated knowledge understandable by humans and to enable human evaluation of WSs' outputs. Explicitly, the contribution presented in this paper attempts to relate both views, consequently enabling machine-human communication: (a) a conceptual view relying on knowledge given by developers to either describe or control the system behavior, and, (b) behavior-related knowledge acquired during WS's learning process. In this way, we use two state-based formalisms:

- STGs for representing behavior-related knowledge generated by the WSs.
- IOSTSs for modelling conceptual views of developers/experts,

The rest of the paper is organized as follows. Section II is dedicated to the context, it presents the basic idea, describes the architectural overview and gives the definition of important terms. Section III presents STG and IOSTS formalisms and illustrates them through examples. Finally, Section IV presents our graph matching algorithm and the construction of two samples of graph matching, one on a roller shutter and the other one on a light bulb to illustrate the algorithm. Section V is devoted to related work before concluding the paper in Section VI.

II. Context & problematic

Reusability and evolution in a wide architectures are two major problems faced when developing applications based on AI. The main challenge is then to provide support for the development of AI-based applications in a way similar to the development of "classical" software using software engineering methods [23][24], tools and languages [25]. We sketch in this section an overall view of how we designed the WO concept and the WOF to support such evolution.

As software systems play an important role in our daily life, their usage may vary depending on the end user and may evolve in time. Our research work is centred on AI-based applications, which are able to acquire data from their environment, to manipulate and to analyze them either to help users take decisions or to autonomously adapt their behavior to users' needs resulting the WOs concept.

A. Basic idea & definitions

The basic idea underlying the WO concept is to give a software entity (object, component, subsystem, etc.) the core mechanisms for learning behavior through introspection and analysis. Our aim is to go further by enabling software to execute "Monitoring", "Analyze", "Plan" and "Execute" loops based on "Knowledge", called MAPE-K [9]. Around this concept, we built the WOF [26] with design decisions mainly guided by reusability and genericity requirements: the framework should be maintainable and used in different application domains with different strategies (e.g., analysis approaches).

Seeking clarity, we have adopted some terms used for humans to refer to abilities a WO possesses. Awareness

and wisdom both rely on knowledge. Inspired by [27], we give some definitions of those terms commonly used for humans [28] and present those we chose for WOs.

Data, it is a raw measurement, typed or not. The data are values obtained from a device or a software component. They represent a quantifiable observation. Generally, collected, stored, processed, transmitted and analyzed, the data have no meaning taken individually. To be relevant, they must be associated with a specific context. In this case, they become information [29].

Information The information provides the data with its context for analysis and processing. The information then becomes understandable by both humans and machines and gives a particular meaning to each data. Information is meaningful and allows us to better answer the questions When? Where? Who? What? etc. The answers to these simple questions do not allow us to make deductions. To do so, it is necessary to go up in abstraction towards knowledge [29]. i.e., placing your hand on a hot stove.

Knowledge: refers to information, inference rules and information deduced from them, for instance: "Turning on a heater will cause temperature change".

Awareness: represents the ability to collect - to provide internal data - on itself by itself. For instance, when an entity/object/device collects information and data about its capabilities *(what is intended to do)* and its use *(what it is asked to do)*. Capabilities are the services/functionalities the WO may render. They are implemented by methods that are invoked by the WO itself during the "dream" phase or from outside during the "awake" phase.

Wisdom: is the ability of WOs to analyze collected information and stored knowledge related to their capabilities and usage to output useful information. It is worth noticing that a WO is highly aware, while the converse is false (an aware object is not necessarily wise).

Semantic: is the subtle shades of meaning given to something so that it can be understood by humans as mentioned in [28]. This definition also applies to objects/devices, as semantic is used to communicate with humans. The value "100" of a variable "data" means nothing to an end user if we do not give her/him the information that it represents a percentage of humidity.

B. WO from an architectural view

From an architectural perspective, according to the target application, a WO may be considered as [9]:

- a stand-alone software entity (object, component, etc.),
- a software avatar designed to be a proxy for physical devices (e.g., a heater, vacuum cleaner, light bulb) [30],
- a software avatar designed to be a proxy for an existing software entity (object, component, etc.).

A WO is characterized by its:

- autonomy: it is able to operate with no human intervention,
- adaptability: it changes its behavior when its environment evolves,
- ability to communicate: with its environment according to a publish-subscribe paradigm.

In addition to the implementation of the MAPE-K loop, another concept, "phase transition", is used to separate the actions performed with a real-world impact (awake phase) from those without (dream phase). As illustrated in Figure 1, three different phases/super-states [8] of the WO [30][26] are defined, the "awake phase", the "dream phase" and the "idle phase".

During the awake phase, a WO responds to different services and requests, i.e., executes its methods called by other objects, monitors its execution and its usages (MAPE-K). Once the WO has finished, it switches to the idle phase. If a part of WO knowledge (e.g., a generated STG: Figure 2) requires analysis or if an AI plugin (see Figure 3) wants to analyze the WO itself, the WO can switch to dream phase. This phase is actually defined as a sub-phase of the idle phase, pushing the WO closer to the human, in other words, when it has nothing to do, it can dream.

Throughout this dream phase, a WO introspects its behavior and analyzes its knowledge without any effect on other objects of the system/real world. The ability of WOs to disconnect from the real world is a strong feature that distinguishes WSs from other self-adaptive systems such as multi-agent systems (MASs) [6]. Furthermore, within this phase, the WO switches from the MAPE-K feedback loop to IAPE-K feedback loop, this latter is nothing more than a technical adaptation of the MAPE-K feedback loop. Thus, rather than monitoring (M) its activity, the WO uses the "introspection" (I) mechanism to discover its behavior and any unusual operation of its use. We will not go into further detail as this topic is beyond the scope of this article.

C. Wise system & the framework

A WS is a set of WOs that communicate their states to the system as illustrated in Figure 3. It highlights a classical WS: a set of instantiated WOs (i.e., Application features, Managers, AIs at the system level) that contains the core (Figure 2) where the basic mechanisms for monitoring are defined. Each WO has three associated components:

- an event communication medium to publish its state to the system,
- a data Logger to log every interaction in/with the object,
- one or more AI plugins to provide the developer with the ability to add objects with different policies of introspection, monitoring, decision and action.

In addition, two kinds of WOs are defined in the system:

- manager: the WS may hold one or more managers (in Orange) that store the peering action/reaction among WOs, using for example Event-Condition-Action (ECA) rules,
- system AI: it manages the whole AI of the system, provided through WOs' AI plugins. AI denotes all activities needed for problem resolution, supervision, learning, analysis. We refer to those activities as Introspect-Monitor-Analyze-Decide-Act (IMADA-activities).

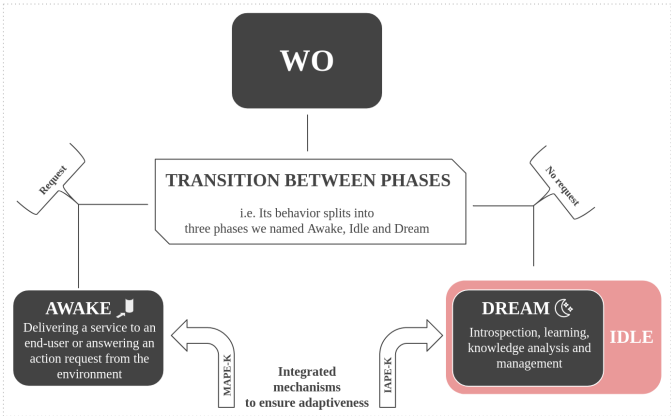


Figure 1. Three different phases/super-states of a WO.

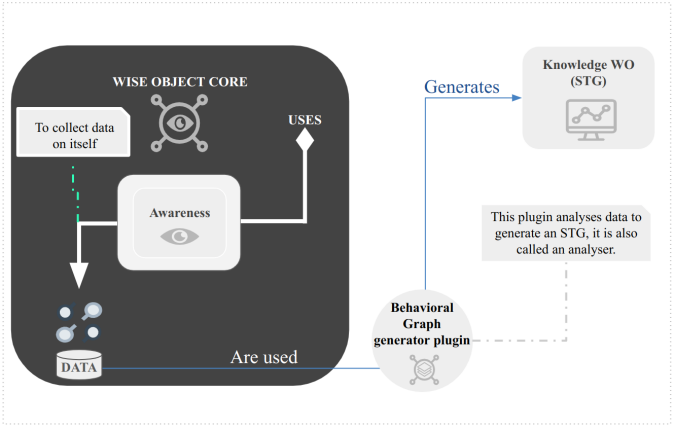


Figure 2. Generic functional architecture of a WO and its relation with an STG.

Regarding the core of the WO, it is designed to be connected to physical devices (e.g., heater, vacuum cleaner, light bulb) or logical entities [9]. It also contains an advanced mechanism to proxify each software entity to make it wise. In addition, the core contains the basic code expressing the two feedback loops (MAKE-k and IAPE-K) accessible to extend by experts depending on the domain/case in which the WO is used.

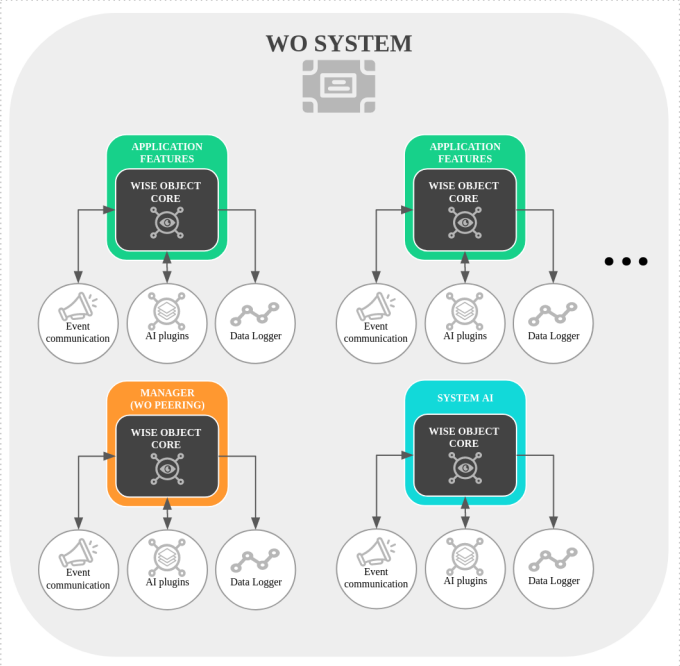


Figure 3. Global view of a WO system composed of a set of WOs, a manager and a system AI.

In order to build WSs, we developed in Java the WOF that provides all the WO mechanisms [31]. It provides the developers with "awareness", "knowledge" and "wisdom" concepts, that are the core. From a system development perspective, the design behind WOF is driven by the following requirements:

- the separation of concerns between the business and AI features,
- minimum intrusion in the business code.

More specifically, the WOF separates the business logic layer from the AI layer, without change in the business layer. Thus, the designers and developers can focus on one hand on the business logic and on the other hand on the AI logic. Details about the WOF is beyond the scope of this article, for more details on the architecture used within the WOF, see [8].

D. The target problem

When designing a WS, developers provide a conceptual model describing, specifying the way they view the behavior of the system's entities associated to WOs. Such models are represented using IOSTS and contain the semantic given by developers to WOs (Section III-B). The IOSTS formalism is mostly known in simplifying system modelling by allowing symbolic representation of parameters and variable values instead of concrete data values enumeration [32]. In addition, and as mentioned in Section II-B, the introspection mechanism helps the WO discover its behavior by using awareness to collect data about itself. These data will be used by the analyzers, for example by the STG generator plugin to build the STG that is a new WO knowledge. This new knowledge can be used by other analyzers like a graph matcher plugin that improves the STG through semantic, using the IOSTS given by an expert or a developer to the WO. This enhancement is based on the graph matching algorithm (Section IV), which will be the main focus of this paper.

This subsection is an implicit description of what is illustrated in Figure 4.

III. Behavioral Models, Definitions and Illustrations

Modeling the behavior of a system is enabled by tools and languages that result in informal, semi-formal (e.g., UML) or formal representations based on already proven theories [33] like graph theory. We have chosen STG and IOSTS graph-based theories to WO's behavior representation, respectively

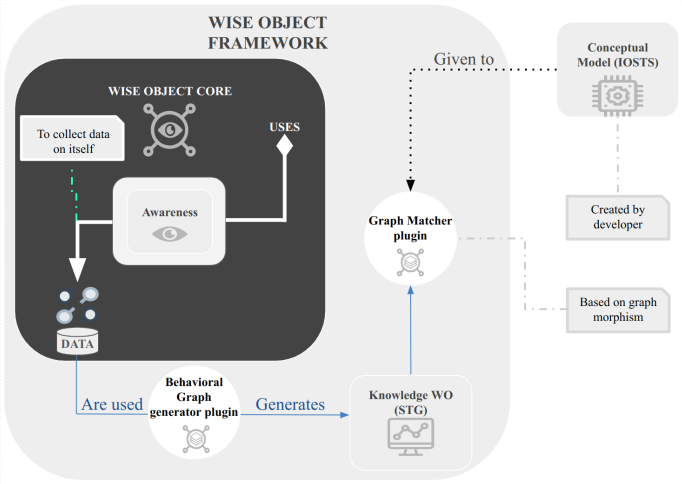


Figure 4. The matching algorithm and the WOF.

at the WO level (i.e., WO's generated view) and the conceptual level (i.e., developer's view).

A. Definition of an STG

An STG is a directed graph where vertices represent the states of an object and transitions represent the execution of its methods. Let us consider an object defined by its set of attributes A and its set of methods M. According to this information (A and M) on the object, the STG definition is given in Definition 1.

***Definition** 1:* An STG is defined by the triplet $G(V, E, L)$ where V and E are, respectively, the sets of vertices and edges, and L a set of labels.

- V is the set of vertices, with $|V| = n$ where each vertex represents a unique state of the object, and conversely, each state of the object is represented by a unique vertex. Therefore $v_i = v_j \Leftrightarrow i = j$ with $v_i, v_j \in V$ and $i, j \in [0, n[$.
- E is the set of directed edges where $\forall e \in E$, e is defined by the triplet $e = (v_i, v_j, m_k)$, such that $v_i, v_j \in V$ and $m_k \in M$. This triplet is called a transition labeled by m_k. The invocation of method m_k from state v_i switches the object to state v_j.
- L is a set of vertex labels where any label $l_i \in L$ is associated to v_i.
  A label l_i is the set of pairs $(att_j, value_{i,j})\ \forall att_j \in A$, with $value_{i,j}$ the value of att_j in the state v_i and $Dom(att_j)$ the value domain of att_j, i.e., the set of $value_{i,j}$ for all i. By definition, 2 states v_i and v_j are different $v_i \neq v_j$, $iff \quad \exists att_k \in A$, such that $value_{i,k} \neq value_{j,k}$. Conversely, if $\forall k \in [0, |A|[\ \ value_{i,k} = value_{j,k}$, the states v_i and v_j are considered the same, i.e., $v_i = v_j$, thus $i = j$.

The matching algorithm we propose in Section IV takes as input an STG with a specific property we name exhaustiveness. The definition of "exhaustive STG" is given in Definition 2.

***Definition** 2:* An exhaustive STG is an STG such that from each vertex v_i there exist $|M|$ transitions, each labeled by a method m_k in M:

$$\forall v_i \in V, \forall m_k \in M, \exists v_j \in V | (v_i, v_j, m_k) \in E.$$

It is worth noting that v_i and v_j may be different or same states ($v_i \neq v_j$ or $v_i = v_j$).

Consequently, an exhaustive STG is deterministic, i.e., from any state, on any method invocation, the destination state is known. Moreover, the number of transitions $|E|$ in an exhaustive STG depends on the number of vertices $|V|$ and methods $|M|$ such that:

$$|V| \times |M| = |E|.$$

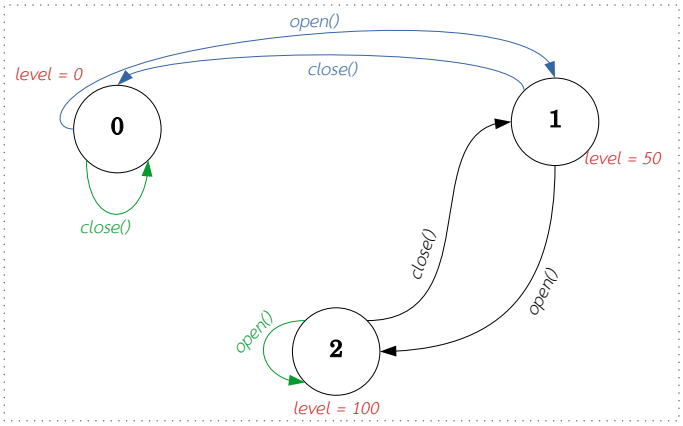


Figure 5. Example of an exhaustive STG.

Figure 5 illustrates an exhaustive STG for an object's behavior, defined by the attribute "level" ($A = \{level\}$) and 2 methods "open" and "close" ($M = \{open(), close()\}$). The methods "open" and "close" increase and decrease the level by 50, respectively. In the STG generated by an object for the shutter, except the methods that give semantic to the transitions, the states have no semantic. Considering the level is initialized to 0, the corresponding STG has 3 states and its exhaustive form has 6 transitions.

B. Definition of an IOSTS

An IOSTS is a directed graph whose vertices, called localities, represent different states of the system (in our case, the system is a software object) and whose edges are transitions. The localities are connected by transitions triggered by actions. In graph theory, an IOSTS allows us the definition of an infinite state transition system in a finite way, contrary to an STG where states are defined by discrete values. IOSTS are used to verify, test and control systems. Verification and testing are formal techniques for validating and comparing two views of a system while control is used to constrain the system behavior [20].

The definition of IOSTS given in Definition 3 is taken from [34][20] and especially from the use case given in [32].

***Definition** 3:* An IOSTS is a sixfold $\langle D, \Theta, Q, q_0, \Sigma, T \rangle$ such as:

- D is a finite set of typed data consisting of two disjoint sets of: variables X and action parameters P. The value domain of $d \in D$ is determined by $Dom(d)$.
- Θ : an initial condition expressed as a predicate on variables X.
- Q is a non-empty finite set of localities with $q_0 \in Q$ being the initial locality. A locality q is a set of states such that $q \in Dom(X)$, with $Dom(X)$ being the cartesian product of the domains of each $x \in X$:

$$Dom(X) = \prod_{\forall x \in X}^{X} Dom(x).$$

  A state is defined by a tuple of values for the whole variables.
- Σ is the alphabet, a finite, non-empty set of actions. It consists of the disjoint union of the set $\Sigma^?$ of input actions, the set $\Sigma^!$ of output actions, and the set $\Sigma^{\mathcal{T}}$ of internal actions. For each action a in Σ, its signature $sig(a) = \langle p_1, \ldots, p_k \rangle | p_i \in P$ is a tuple of parameters. The signature of internal actions is always an empty tuple.
- T is a finite set of transitions, such that each transition is a tuple $t = \langle q_o, a, G, A, q_d \rangle$ defined by:
  - a locality $q_o \in Q$, called the origin of the transition,
  - an action $a \in \Sigma$, called the action of the transition,
  - a boolean expression G on $X \cup sig(a)$ related to the variables and the parameters of the action, called the transition guard. Transition guards allow us to distinguish transitions that have the same origin and action but disjoint conditions to their triggering.
  - An assignment of the set of variables, of the form $(x := A^x)_{x \in X}$ such that for each $x \in X$, A^x is an expression on $X \cup sig(a)$. It defines the evolution of variable values during the transition,
  - a locality q_d, called the transition destination.

According to Definition 3, each variable has a subdomain in each locality. Thus, let us define the function $dom(q, x)$ that returns the definition domain of the variable $x \in X$ in the locality $q \in Q$; consequently $dom(q, x) \subseteq Dom(x)$.

Figure 6 shows an example of an IOSTS given by a developer to control a roller shutter. This IOSTS expresses that the roller shutter expects an input $up?/down? \in \Sigma^?$ carrying the parameter $step \in]0, 100]$, the relative elevation to respectively increase or decrease the shutter level. Let us note that the shutter elevation is between 0 and 100.

There are 2 localities:

- The locality where the system is closed (i.e., $height = 0$). If the system receives the $up?(step)$ command, the transition will be made from the $Closed$ to $Open$ locality by increasing the value of the $height$ variable by $step$, but if the system receives the $down?(step)$ action, it will not perform any operation (NOP).
- The locality where the system is open (i.e., $height \in]0, 100]$). If the system receives the action $up?(step)$, the transition will be reflexive from $Open$ to itself and will compute the value of the variable $height$ by executing this assignment $height = min(height + step, 100)$, the shutter elevation cannot be increased more than the maximum of elevation. If it receives the $down?(step)$ action and the action closes the shutter less than it is open ($step < height$), $height$ is decreased by $step$, otherwise the transition will be from the locality $Open$ to the locality $Close$ by assigning 0 to the variable $height$.

According to Definition 3, this IOSTS is composed of the sets of variables $X = \{height\}$ with $Dom(height) \in [0, 100]$ and parameters $P = \{step\}$ with $Dom(step) \in]0, 100]$, the set of localities $Q = \{Open, Closed\}$ and the set of actions $\Sigma = \{up?, down?\}$ where the signatures of the actions are $sig(up?) = sig(down?) = \langle step \rangle$. This IOSTS models an infinite state system based on 5 guarded transitions in T:

$$\begin{aligned} T = \quad \langle \quad & t_{Close-Open}, \\ & t_{Open-Close}, \\ & t^1_{Open-Open}, \\ & t^2_{Open-Open}, \\ & t_{Close-Close} \quad \rangle \end{aligned}$$

such as:

$$\begin{aligned} t_{Close-Open} = \quad \langle \quad & Open, up?(step), \\ & True, height := height + step, \\ & Open \rangle \\ t_{Open-Close} = \quad \langle \quad & Open, down?(step), \\ & step \geq height, height := 0, \\ & Close \rangle \\ t^1_{Open-Open} = \quad \langle \quad & Open, down?(step), \\ & step < height, height := height \\ & -step, Open \rangle, \\ t^2_{Open-Open} = \quad \langle \quad & Open, up?(step), \\ & True, min(height + step, 100), \\ & Open \rangle, \\ t_{Close-Close} = \quad \langle \quad & Close, down?(step), \\ & True, NOP, \\ & Close \rangle. \end{aligned}$$

As can be noticed, there exists an infinity of paths and states represented by the variable $height$ since its domain is the interval $[0, 100]$.

IV. Graph matching algorithm

In this section, we introduce the matching algorithm we propose to relate WO's generated STG to developers' semantic expressed in an IOSTS. In the example of Figure 5, the generated STG is composed of states automatically labelled by the object: 0, 1 and 2 according to the value of attribute level: 0, 50, 100. The main challenge is how to match states 0, 1 and 2 to the localities defined by developers in the IOSTS of Figure 6.

A. Matching algorithm

Constraint: The STG and IOSTS must meet certain criteria to properly apply the algorithm.

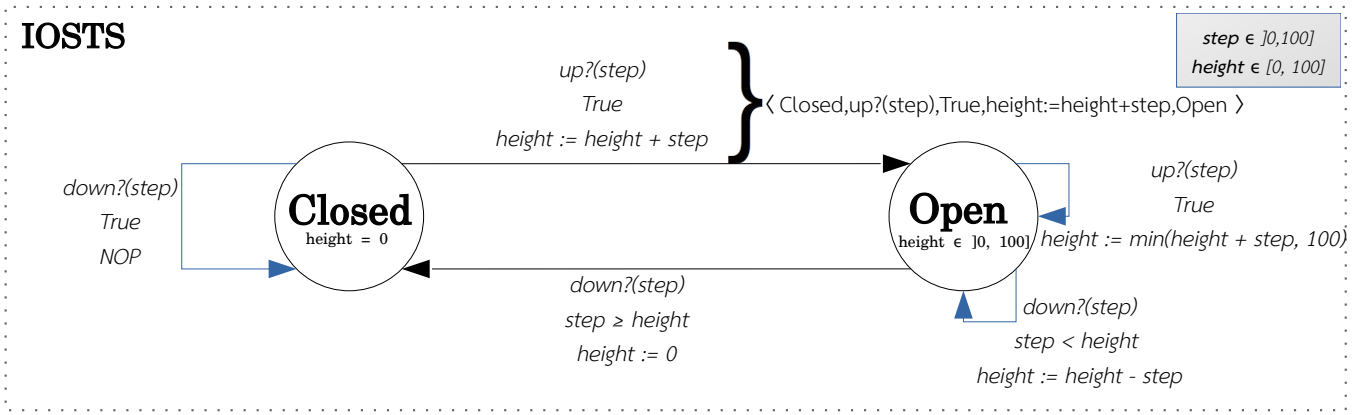


Figure 6. IOSTS representation of a roller shutter.

1) There are two equivalent characteristics: a variable x_e belonging to the set of variables X of the IOSTS and an attribute att_e belongs to the set of STG attributes A. Moreover, to simplify the problem in this paper, let us consider they are unique:

$$\exists! x_e \in X, \exists! att_e \in A | x_e \equiv att_e, \tag{1}$$

$x_e \equiv att_e$ means that both represent the same information, thus:

$$Dom(att_e) \subseteq Dom(x_e). \tag{2}$$

Let us note that $Dom(att_e)$ is a subset of $Dom(x_e)$ due to the fact that x_e is theoretically defined into the IOSTS and att_e is partially discovered by the WO at runtime.

2) The domains of x_e in the different localities in the IOSTS are disjoint:

$$\forall q, q' \in Q, q \neq q' \Leftrightarrow dom(q, x_e) \cap dom(q', x_e) = \emptyset$$

Algorithm: According to the definitions of STG and IOSTS, and both constraints, a vertex $v_i \in V$ matches a locality $q_j \in Q$ (noted $v_i \implies q_j$) if and only if $value_{i,e} \in dom(q_j, x_e)$, with $value_{i,e}$ the value of att_e in the vertex v_i:

$$\begin{array}{c} \forall v_i \in V, \exists! q_j \in Q \\ value_{i,e} \in dom(q_j, x_e) \Leftrightarrow v_i \implies q_j. \end{array} \tag{3}$$

As the matching algorithm is a graph morphism, this latter needs to respect the structure of the matched graphs [35]. In our context, each vertex matches one locality and a locality is matched by at least one vertex. Moreover, the adjacency relations must be respected by the matching; if 2 vertices are linked by a transition in the STG, their matched localities are the same or linked by an equivalent transition in the IOSTS. The STG $\rightarrow$ IOSTS matching is a surjective $\mathcal{S}$ homomorphism, i.e., epimorphism [35] as illustrated in the formulas below:

$$\begin{array}{rl} \mathcal{S}\colon STG & \rightarrow IOSTS \\ V & \twoheadrightarrow \mathcal{S}(V) = Q, \end{array} \tag{4}$$

implies:

$$\mathcal{S}_E\colon E \rightarrow T,\ \mathcal{S}_E((v_i, v_j)) = (\mathcal{S}(v_i), \mathcal{S}(v_j)) \subset T. \tag{5}$$

For any transition $(v_i, v_j) \in E$ of STG, then $(\mathcal{S}(u_i), \mathcal{S}(v_j)) \in T$ is a transition of the IOSTS.

B. Stepwise matching algorithm

In this section, the matching algorithm will be presented step by step, as illustrated in Figure 7, to understand deeply how it works. The algorithm is divided into two steps, the first matches the attribute-variable according to their domain definition and defined by Function "$compatibleDomain$". The second step is devoted to compute the matching between states and localities, which implementation is described in Algorithm 1 (main algorithm). Furthermore, Algorithm 2 is the implementation details of the "$compatibleDomain$" function whose result will be part of the inputs to the main algorithm.

In detail, equations (1) and (2) are developed in the "$compatibleDomain$" function, which is exposed in algorithm 2. And Equation (3) is exposed in algorithm 1. This pseudo-code is the first version of the matching algorithm we have developed and tested as a first step towards human semantic.

C. First matching illustration

In the previous examples: the STG in Figure 5 is automatically generated by a WO and the IOSTS in Figure 6 is provided by a developer. Both represent the same roller shutter behavior. The STG uses discrete values with a level of opening of 50%, while the IOSTS uses continuous intervals, without any constraint on the step that is a real value.

Figure 8 illustrates the result of matching both graphs using our graph matcher implemented with Python. Localities in the IOSTS are $Closed$ and $Open$, each containing variables with disjoint domains, in our example, a single variable named $height$ that takes different values depending on its locality.

According to the constraints of the matching algorithm:

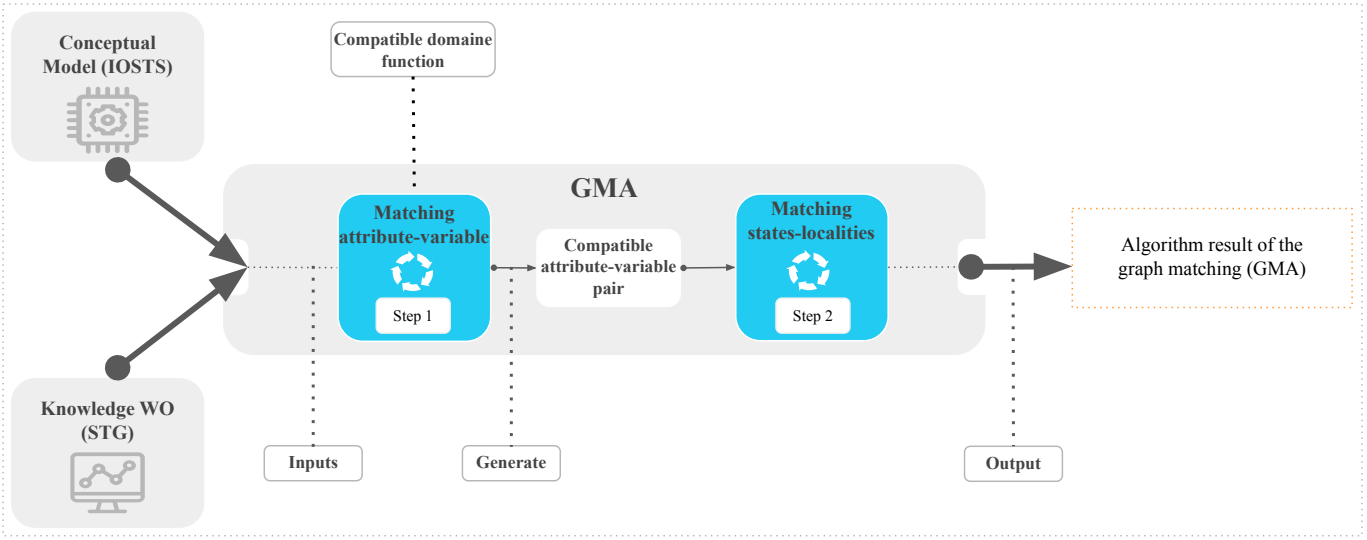


Figure 7. Illustration of stepwise matching algorithm applied on roller shutter (output in Figure 8).

1) there are equivalent characteristics between the STG and the IOSTS, the attribute "level" and the variable "height", respectively,
2) the domains of "height" in the different localities are disjoint from the others: in $Closed$ locality, the variable can only take the value 0 and in $Open$ locality, the variable can take any value in the range $]0, 100]$.

On the STG side, there are three vertices, each one labeled with a set of attribute-value pairs (att, value). In our case, the unique attribute $level$ takes the values $(0, 50, 100)$ respectively for (v_0, v_1, v_2). Therefore, to establish a correspondence between the two graphs, a comparison between the definition domain of the attribute $level$ in each vertex of the STG with the definition domain of the variable $height$ in each locality of the IOSTS must be done.

Those comparisons lead us to a correspondence of state v_0 with locality Closed meaning that the roller shutter is closed, and a correspondence of states v_1 and v_2 with locality Open meaning that the roller shutter is open.

D. Second matching illustration

The second illustration concerns a connected light bulb with RGB colours. To simplify the illustration, only two colors are considered and the off state of the light bulb is not considered. Figure 9 shows both simplified behavioural graphs of the light bulb: STG and IOSTS. The former is defined by the attribute "specter" ($A = \{specter\}$) and 2 methods "upFrequence" and "downFrequence" ($M = \{upFrequence(), downFrequence()\}$). The "upFrequence" and "downFrequency" methods increase and decrease the specter by 40nm, respectively. The second graph contains two localities $Green$ and $Blue$, each containing variables with disjoint domains, in our example, a single variable named $wavelength$ that takes different values depending on its locality.

The constraints of the matching algorithm are respected:

1) there are equivalent characteristics between the STG and the IOSTS, the attribute "specter" and the variable "wavelength", respectively,
2) the domains of "wavelength" in the different localities are disjoint from the others: in $Green$ locality, the variable can take any value in the range $]495, 570]$ and in $Blue$ locality, the variable can take any value in the range $[450, 495]$.

The algorithm lead us to a correspondence of states v_0, v_1 with locality $Blue$ meaning that the bulb is in blue color, and a correspondence of states v_2, v_3 with locality $Green$ meaning that the bulb is in green color.

V. Related work

For many years, graphs have been used in several fields to represent complex problems in a descriptive way (e.g., maps, relationships between people profiles, public transportation, scene analysis, chemistry, molecular biology, and so on) for various purposes: analysis, operation, knowledge modeling, pattern detection, etc. Although initiated in the 18th century with Euler's work on the famous problem of Königsberg bridges [36], graph theory remains a powerful tool for software-intensive system development and an effective way of representing objects as proved in [37]. Since then, several approaches of graph matching have been developed and the first formulation of the graph matching problem was proposed by [38] and dates back to 1979. Several formulations appeared afterwards like convex-concave programming formulation, maximum common subgraph (MCS), the use of the Frobenius norm based on graph adjacency matrices to express the maximization or the minimization of non-overlapping edges between two graphs. In general, there exist two major formulations for the graph matching problem [39][40]:

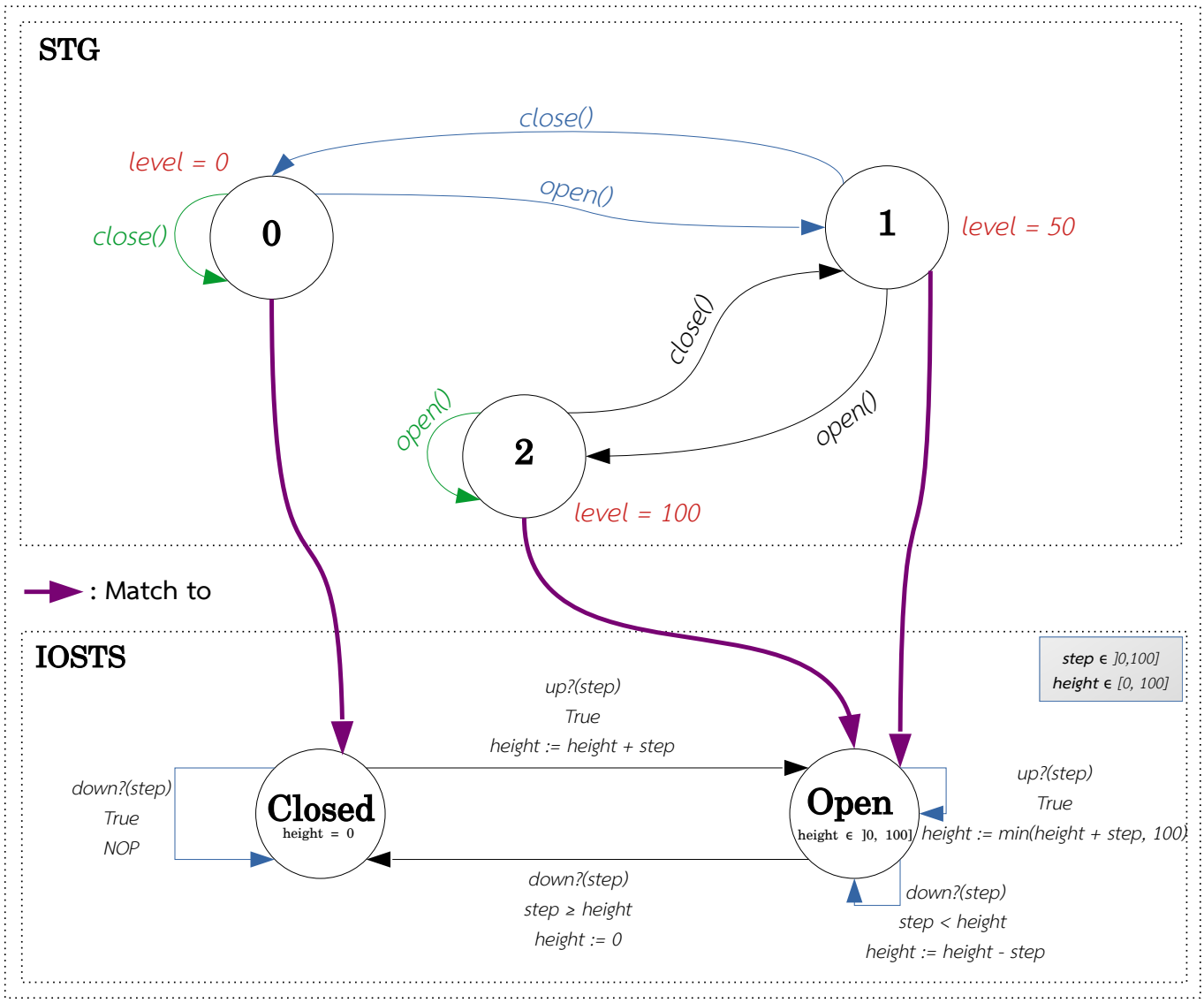


Figure 8. Algorithm result of the graph matching (roller shutter) [1].

- Exact Matching which is divided into two categories, (a) graph isomorphism, checks whether two graphs are the same. (b) subgraph isomorphism, checks whether the smallest graph is a subgraph of the biggest one. Both techniques are overly complex whether or not they check the one-to-one or many-to-one matching.
- Inexact Matching which is a term used in the case where it is impossible to find an isomorphism between two graphs. This form of matching is based on several approaches:
  - the maximum common subgraph, used in searching the similarity between graphs to know how different they are instead of a binary answer [41].
  - least-squares formulation, used in the case of weighted graphs to search for a matching that minimizes the total difference between all aligned edges through the use of the Frobenius norm for instance [41].
  - graph edit distance, used to find in a low cost the sequence of operations (i.e., deletion, insertion and substitution of vertices and edges) that transform one graph into another [42]. As this procedure is a hard combinatorial problem, another alternative called “beam search” is explained in details in [43].

In real applications, we often wish to match graphs of different sizes, which results in new techniques and norms as depicted in [39]. Moreover, as many formalisms have emerged so far, the correspondence between different representations of knowledge such as STGs and IOSTSs, has not been addressed yet at the best of our knowledge. Until now, the most well-known operation on graphs is the comparison of two or more graph representations that requires many theoretical

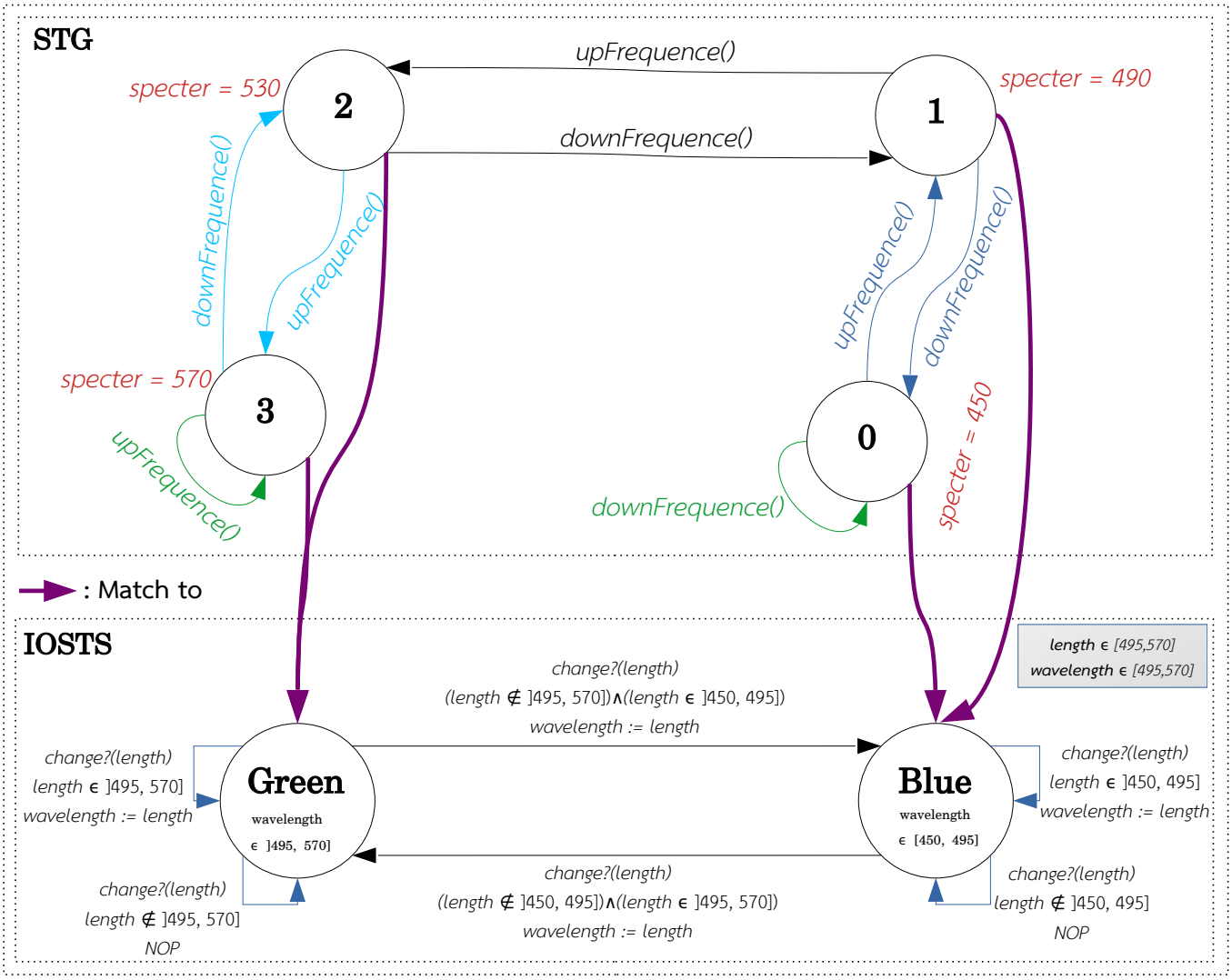


Figure 9. Algorithm result of graph matching (light bulb).

and complex concepts [21], like graph matching, a noisy version of graph isomorphism that is at the basis of our proposal in this paper. Finally, we mention that graph/sub-graph isomorphism is considered the most complex problem in graph matching as it has been proven to be NP-complete in [44][45][46] Moreover, for certain types of graphs under given constraints, the complexity of the isomorphism has been proven of polynomial type with a huge cost [47].

Using the constraints presented in Section IV to match two knowledge representations (STG - IOSTS) lead us to the exact matching problem. To understand the situation, we need to consider the matching from both perspectives: software (i.e., numerical and structural) and human (i.e., semantic). According to the software, and since the matching preserves the structure and the transitions between both formalisms, the matching is always exact between "states" and "localities", which gives an epimorphism (Equations (4) and (5)). From a human perspective, we will always have an exact matching based on semantic as illustrated in Figures 8 and 9. The question is how to match STGs and IOSTs when constraints are expanded to include more than one equivalent attribute-variable. In this case, we should adopt an inexact matching approach so that the algorithm generates more than one result (see future work for more details).

VI. Conclusion and Future Work

In this paper, we addressed the problem of relating numerical representations generated by WSs to developers' semantic. The contribution is a matching algorithm that computes a morphism between two behavioral graphs:

1) an STG generated by a WO along its learning process,
2) an IOSTS representing a developer conceptual view.

The algorithm extends a WO's view with semantic that allows it to communicate with humans. From the developer's perspective, the resulted matching may help him/her discover errors and/or inconsistencies between the conceptual view and

Algorithm 1 Main algorithm: Graph matching algorithm

Inputs:
  $iosts$: IOSTS,
  stg: Exhaustive STG
Outputs:
  $match$: Dictionary<state, locality>
Locales:
  # Set of possible attribute/variable pairs
  E: Set Of Tuples< (attribute, variable) >
  # Possible matches for each possible equivalent pair
  M: Dictionary< (attribute, variable), <state, locality>>
Initialize:
  # Build possible equivalent attribute/variable pairs, such that $dom(a) \subseteq dom(x)$ (Algorithm 2)
  $E \leftarrow compatibleDomain(stg.A, iosts.X)$
for $(a, x) \in E$ **do**
  for $v_i \in stg.V$ **do**
# Get the locality where the domain of variable x contains the value of a in v_i, according to the second constraint, q_i is unique
    $q_i \leftarrow iosts.getLocality(x, v_i.getValue(a))$
    $M((a, x))(v_i) \leftarrow q_i$
  end for
# As the matching is a surjective application, remove the pair if it does not generate surjective matching
  if $M((a, x)).getKeys() \neq stg.V$
  or $M((a, x)).getValuesAsSet() \neq iosts.Q$ **then**
    E.remove((a,x))
    M.remove((a,x))
  else
# If the application is surjective (Equation (4)), check the transitions' consistency (Equation (5))
    for $e \in stg.E$ **do**
      $v_1 \leftarrow e.getSource()$
      $v_2 \leftarrow e.getDestination()$
      $q_1 = M((a, x))(v_1)$
      $q_2 = M((a, x))(v_2)$
      if $iosts.getTransition(q_1, q_2)$ **is null then**
        E.remove((a,x))
        M.remove((a,x))
      end if
    end for
  end if
end for
# Checking that just only one matching exists according to constraints defined in Section IV-A
if $M.getKeys().size() == 1$ **then**
  $match \leftarrow M.getValues()[1]$
else
  $exception$("Required conditions not satisfied")
end if
return $match$

Algorithm 2 Variable matching algorithm ("compatibleDomain" function)

Inputs:
  $iosts$: IOSTS,
  stg: Exhaustive STG
Outputs:
  # Set of possible attribute/variable pairs such that $Dom(attribute) \subseteq Dom(variable)$
  E: Set Of Tuples< (attribute, variable) >
# Build all possible equivalent attributes-variables using the cartesian product between A and X
for $cartesian \in product(stg.A, iosts.X)$ **do**
# Keep attribute-variable pairs, such that $Dom(a) \subseteq Dom(x)$
  if $Dom(cartesian.getAttribute()) \subseteq Dom(cartesian.getVariable())$ **then**
    $E.add(cartesian)$
  end if
end for
return E

the system implementation. In its first version, the algorithm has obviously several limitations, the strongest being over the number of equivalent attributes/variables in STG/IOSTS. Another limitation is the constraint on the existence of only one matching between an STG and an IOSTS, without omitting the problem of inexact matching. Ongoing work is being done to gradually generalize the algorithm and raise those restrictions. The graph matching algorithm being an NP-complete problem, we envisage the use of ontologies in a matrix form through two main matrices, termed "Semantic Matrix" and "Graph Matching Matrix". Moreover, we initiated a France-Canada innovation project to apply our approach to help create assistive scenarios [10][11] for elderly people, in the context of smart home.

Acknowledgment

This research was supported by French National Research Agency (ANR), AI Ph.D funding project.

References

[1] A. Dahhani, I. Alloui, S. Monnet, and F. Vernier, "Towards a semantic model for wise systems: A graph matching algorithm," Proceedings of the Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2022), IARIA, 2023, pp. 27–34.

[2] E. Nascimento, A. Nguyen-Duc, I. Sundbø, and T. Conte, "Software engineering for artificial intelligence and machine learning software: A systematic literature review," 2020.

[3] R. Feldt, F. G. D. O. Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," 2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), pp. 35–41, 2018.

[4] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in International Conference on Software Engineering (ICSE 2019) - Software Engineering in Practice track, May 2019. ICSE 2019 Best Paper Award.

[5] E. Kusmenko, S. Pavlitskaya, B. Rumpe, and S. Stüber, "On the engineering of ai-powered systems," in 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), pp. 126–133, 2019.

[6] R. A. Flores-Mendez, "Towards a standardization of multi-agent system framework," XRDS, vol. 5, pp. 18–24, jun 1999.

[7] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," IEEE Access, vol. 6, pp. 28573–28593, 2018.

[8] S. Lejamble., I. Alloui., S. Monnet., and F. Vernier., "A new software architecture for the wise object framework: Multidimensional separation of concerns," in Proceedings of the 17th International Conference on Software Technologies - ICSOFT, pp. 567–574, INSTICC, SciTePress, 2022.

[9] I. Alloui and F. Vernier, "WOF: Towards Behavior Analysis and Representation of Emotions in Adaptive Systems," Communications in Computer and Information Science, vol. 868, pp. 244–267, 2018.

[10] D. Bonino and F. Corno, "Dogont - ontology modeling for intelligent domotic environments," in The Semantic Web - ISWC 2008, pp. 790–803, Springer Berlin Heidelberg, 2008.

[11] H. Kenfack Ngankam, H. Pigot, M. Frappier, C. H. Souza Oliveira, and S. Giroux, "Formal specification for ambient assisted living scenarios," UCAmI, pp. 508–519, 2017.

[12] J.-B. Woo and Y.-K. Lim, "User experience in do-it-yourself-style smart homes," in Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 779–790, 2015.

[13] R. Radziszewski, H. Ngankam, H. Pigot, V. Grégoire, D. Lorrain, and S. Giroux, "An ambient assisted living nighttime wandering system for elderly," in Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, iiWAS '16, pp. 368–374, Association for Computing Machinery, 2016.

[14] R. S. Michalski, "A theory and methodology of inductive learning," in Machine Learning: An Artificial Intelligence Approach, pp. 83–134, Springer Berlin Heidelberg, 1983.

[15] M. Weiser, "The computer for the 21st century," Scientific American, vol. 265, no. 3, pp. 66–75, 1991.

[16] M. Weiser and J. S. Brown, "Designing calm technology," PowerGrid Journal, vol. 1, pp. 75–85, 1996.

[17] A. Tugui, "Calm technologies in a multimedia world," Ubiquity, vol. 2004, pp. 1–5, 2004.

[18] I. Alloui and F. Vernier, "A Wise Object Framework for Distributed Intelligent Adaptive Systems," in ICSOFT 2017, the 12th International Conference on Software Technologies, pp. 95–104, 2017.

[19] C. Constant, T. Jéron, H. Marchand, and V. Rusu, "Validation of Reactive Systems," in Modeling and Verification of Real-TIME Systems - Formalisms and software Tools, pp. 51–76, Hermès Science, 2008.

[20] C. Constant, T. Jéron, H. Marchand, and V. Rusu, "Integrating Formal Verification and Conformance Testing for Reactive Systems," IEEE Transactions on Software Engineering, vol. 33, no. 8, pp. 558–574, 2007.

[21] M. R. Garey and D. S. Johnson, "Computers and intractability. a guide to the theory of np-completeness.," Journal of Symbolic Logic, vol. 48, no. 2, pp. 498–500, 1983.

[22] V. A. Cicirello, "Survey of graph matching algorithms," technical report, Geometric and Intelligent Computing Laboratory, Drexel University, 1999.

[23] B. Kitchenham, "A methodology for evaluating software engineering methods and tools," in Experimental Software Engineering Issues: Critical Assessment and Future Directions (H. D. Rombach, V. R. Basili, and R. W. Selby, eds.), (Berlin, Heidelberg), pp. 121–124, Springer Berlin Heidelberg, 1993.

[24] B. W. Boehm, "Software engineering - as it is," IEEE Trans. Computers, vol. 25, no. 12, pp. 1226–1241, 1976.

[25] D. Torre, M. Genero, Y. Labiche, and M. Elaasar, "How consistency is handled in model-driven software engineering and UML: an expert opinion survey," Software Quality Journal, pp. 1–53, Apr. 2022.

[26] I. Alloui, D. Esale, and F. Vernier, "Wise objects for calm technology," in Proceedings of the 10th International Conference on Software Engineering and Applications - ICSOFT-EA, (ICSOFT 2015), pp. 468–471, INSTICC, SciTePress, 2015.

[27] T. Davenport and L. Prusak, Working Knowledge: How Organizations Manage What They Know, vol. 1. Harvard Business School Press, 1998.

[28] "Cambridge Dictionary Online," 2022.

[29] H. K. Ngankam, Modèle Sémantique d'Intelligence Ambiante pour le Développement Do-It-Yourself d'Habitats Intelligents. Theses, Faculté des sciences, université de sherbrooke, 2019.

[30] I. Alloui, E. Benoit, S. Perrin, and F. Vernier, "Wise objects for IoT (WIoT): Software framework and experimentation," Communications in Computer and Information Science, pp. 349–371, 2019.

[31] I. Alloui, E. Benoit, S. Perrin, and F. Vernier, "Wiot: Interconnection between wise objects and iot," in ICSOFT 2018, the 13th International Conference on Software Technologies, 2018.

[32] P. Moreaux, F. Sartor, and F. Vernier, "An effective approach for home services management," in 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 47–51, 2012.

[33] M. N. Nicolescu and M. J. Matarić, "Extending behavior-based systems capabilities using an abstract behavior representation," in AAAI 2000, pp. 27–34, 2000.

[34] V. Rusu, H. Marchand, and T. Jéron, "Automatic verification and conformance testing for validating safety properties of reactive systems," in Formal Methods 2005 (FM05), vol. 3582 of Lecture Notes in Computer Science, pp. 189–204, Springer-Verlag, 2005.

[35] G. Hahn and C. Tardif, "Graph homomorphisms: structure and symmetry," in Graph Symmetry: Algebraic Methods and Applications, pp. 107–166, Springer Netherlands, 1997.

[36] H. Sachs, M. Stiebitz, and R. Wilson, "An historical note: Euler's königsberg letters," Journal of Graph Theory, vol. 12, pp. 133 – 139, 2006.

[37] M. A. Eshera and K.-S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, pp. 604–618, 1986.

[38] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 12, pp. 757–768, 1979.

[39] M. Zaslavskiy, L'alignement de graphes : applications en bioinformatique et vision par ordinateur. Theses, École Nationale Supérieure des Mines de Paris, Jan. 2010.

[40] E. Bengoetxea, Inexact Graph Matching Using Estimation of Distribution Algorithms. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

[41] J. R. Ullmann, "An algorithm for subgraph isomorphism," J. ACM, vol. 23, pp. 31–42, jan 1976.

[42] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," Pattern Recognition Letters, vol. 1, no. 4, pp. 245–253, 1983.

[43] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in Structural, Syntactic, and Statistical Pattern Recognition, (Berlin, Heidelberg), pp. 163–172, Springer Berlin Heidelberg, 2006.

[44] D. A. Basin, "A term equality problem equivalent to graph isomorphism," Information Processing Letters, vol. 51, no. 2, pp. 61–66, 1994.

[45] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP - completeness. W.H. Freeman and Co., 1979.

[46] M. A. Abdulrahim, Parallel algorithms for labeled graph matching. Colorado School of Mines1500 Illinois St. Golden, CO, 1998.

[47] J. E. Hopcroft and J. K. Wong, "Linear time algorithm for isomorphism of planar graphs (preliminary report)," in Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC '74, (New York, NY, USA), pp. 172–184, Association for Computing Machinery, 1974.

Preparing Students for the Software Industry New Demands

José Carlos Metrôlho[1,2], Fernando Reinaldo Ribeiro[1,2], Rodrigo Batista[2]
[1]R&D Unit in Digital Services, Applications and Content
[2]Polytechnic Institute of Castelo Branco
Castelo Branco, Portugal
e-mail: metrolho@ipcb.pt, fribeiro@ipcb.pt, rodrigo.batista@ipcbcampus.pt

Paula Graça
DEETC of Instituto Superior de Engenharia de Lisboa
Instituto Politécnico de Lisboa
Lisbon, Portugal
e-mail: paula.graca@isel.pt

Diogo Pacheco
Do iT Lean
Leiria, Portugal
e-mail: diogo.pacheco@doitlean.com

***Abstract*—A solid preparation in terms of soft skills and state-of-the-art technical skills in Software Engineering (SE) is a goal for the academy. It also contributes to reducing the gap between Software Engineering education and the software industry's new demands. Generally, in computer science or computer engineering courses, there are separate subjects to teach requirements engineering, analysis, design, coding, or validation. However, integrating all these subjects usually requires experience in developing a complete project. This article describes aspects of an active and collaborative learning approach involving academia and industry actors. The approach presented in this article involved staff from a software company in collaboration with staff from an academic institution. It resulted in a student being involved in an entire software development project. The student was involved in an agile team of faculty and Information Technology (IT) professionals. The Scrum agile framework was followed, and the product was developed using a Low-code development platform. This article presents the approach, details of the project design and implementation, results achieved, lessons learned, and guidelines for the future. The results show that this agile, full-stack approach allows students to develop cutting-edge technical and non-technical skills.**

Keywords- agile software development; cognitive services; form recognizer; Scrum; software engineering; software industry.

I. INTRODUCTION

Preparing students' performance and technical skills for the software industry's new demands is challenging. If, on the one hand, they must have deep knowledge of specific technical subjects (databases, programming languages, requirements analysis, Web development, mobile development.), it is also increasingly important that they have the skills to integrate or explore features in more complex systems. This broader view of specific software ecosystems requires a well-prepared new generation of engineers using new approaches and a more holistic experience of modern software development activity. These approaches can accelerate development performance and obtain better-designed and high-quality software products.

In the software industry, many advances are also happening to speed up development. Examples of this are the low-code development platforms, which provide an abstraction layer that allows the developers to handle more of the inherent complexity of application development and simultaneously explore reuse and integrate different frameworks. They allow fast learning development processes, enable a more systemic view of software projects, and provide easy integration with other application endpoints. However, SE gains importance here because its inherent abstraction requires following good development practices.

Another essential aspect nowadays is the high possibility of integration and interconnection between various systems. This aspect makes it increasingly crucial for new IT professionals to know the services available and what mechanisms to integrate them into their applications. This holistic knowledge can be acquired in theory, but nothing is better than consolidating it through developing projects that use this integration and other technologies. Cloud service providers (Amazon Web Services, Microsoft® Azure Cloud Platform, or Google Cloud Platform) are cases in point.

In a previous article [1], we share a case study that aims to prepare students for this reality, still in the academic environment, and close collaboration with partners from the software development industry. We described an overview of our approach to prepare better students for the labor market in collaboration with software industry members. In this work, we also approached the importance of full-stack development and preparing students for the entire spectrum of full-stack technologies. The job market needs more technically well-prepared graduates with good soft skills. Thus, preparing the new generation of engineers requires training not only in the technical subjects that are the knowledge base but also the vision and more holistic experience about the paths followed today by software companies and soft skills. These aspects can be tackled with strategies and case studies like the one presented in this article. The product developed in this case was an application for household accounting, automatically recognizing data from existing invoices in digital format (pdf, photo) using cognitive services.

In this article, we extend our description of the approach, including more detailed technical issues considered in this

case study, to foster an embracing and holistic preparation of the students in this learning ecosystem. To achieve this goal, we went beyond the usual development of an academic project. We included issues about security, UI/UX, performance, data synchronization, or integration with external cognitive services.

In this case study, an important fact was that a company that develops software for the international market was involved. This collaboration helps to prepare students for the software industry demands, and with a holistic point of view about SE, involving several aspects that are important to foster effective implementations of active learning (allowing students to acquire knowledge in a practical way) and collaborative learning (creating environments that emulate as much as possible future work environments, whereas software engineers will work together to produce software products). The company defined the product/goal. A student from a higher education institution (academy), integrated into a distributed team, developed it, using Scrum [2] as a software development process. This combination of several contributions and developing a full-stack project using an agile software development process allows the student to acquire the knowledge and preparation necessary for today's challenges in the modern competitive software development market. The main goal is to contribute to reducing the gap that sometimes exists between what is learned in academia and what is needed in the industry. In this article, based on the experience observed in a successful case, we share an approach to prepare students for the software industry competently. The results were very positive, both in technical terms (software product and scope of the student's technical skills) and SE terms (student preparation in terms of useful soft skills for the labor market).

The remainder of this article is organized as follows. Section II presents a background and related work. In Section III, the case study is presented. In Section IV, results and discussion about lessons learned are presented. Finally, some conclusions are presented in Section V.

II. BACKGROUND

Developers often do not just play a single role in software development; they must be multifaceted, often taking on the role of designers, coders, and database specialists. Therefore, having this knowledge and multi-tasking skills is essential and allows the developer to use them to complete a project or software development independently. This is also an advantage because it will enable the developer to be more familiar with all stages of the development process, making cooperation inside and outside the team more optimized and contributing to reducing software development costs. These professionals should be able to work both on Web and mobile platforms with also knowledge of design through the Web, like Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS). In addition, they should be able to use software development tools and techniques that allow the development team to be at its highest level of productivity.

However, higher education institutions face challenges in preparing students to work proactively in these high-performance teams. Many approaches have been proposed to teach and learn SE subjects. Some attempt to motivate students to take a more active role in their training and provide them with more realistic experiences by replicating the settings used in the software development profession has been made. Project-based Learning (e.g., [3]), flipped classroom (e.g., [4]), and gamification (e.g., [5], [6]) are some of these strategies that are frequently used to teach SE. Some other strategies promote a closer involvement of software companies to reduce the gap between SE education and the needs and practice in the software industry. For instance, in the approach described in [7], the industry actively supervises software product development. Another approach is to create additional training programs that aid in screening qualified candidates, as presented in [8]. These approaches are essential for students because they provide real challenges, more realistic experiences, and recreating industry software development environments. Nevertheless, they are also crucial for companies. For them, potential advantages are networking with students and other corporate sponsors, building ties with faculty, and promoting their business and products among college students.

From a different perspective, SE teaching has adapted to new developments and trends, namely agile methodologies. Frequently, teaching agile methodologies has focused on teaching a specific framework like Scrum (e.g., [9]–[11]) or Extreme Programming (e.g., [12], [13]). A study on using Agile Methods in SE Education [14] concluded that using Agile practices would positively influence the teaching process, stimulating communication, good student relationships, active team participation, and motivation for present and future learning.

Besides the good results of several of these strategies, SE teaching and learning can still benefit from a more participative and closer involvement of software development companies in the training process. This participation and involvement enable students to join distributed teams, enhance their non-technical skills, and engage in the practices used in these companies.

III. THE CASE STUDY

A. Context

The work described here was done by a student who attended a computer engineering course's third curricular year (fifth and sixth semesters). In parallel, the student had to attend other curricular units, which are part of the course. The final evaluation of the work done by the student, in academic terms and for diploma purposes, was done by a jury composed of a supervisor (one of those who followed the work) and two teachers external to the project. As for the role of the others involved in this approach, information concerning this is presented in section C.

B. The Elicitation Stage

At the beginning of the course/project, the start of the first semester, the student was guided to be prepared for research and analysis skills. This guidance happened briefly before Sprint 1 (as presented in the timeline in section D). He was led to research and analyze the results of a search in scientific

citation libraries and online sources related to the focus of the application/product goals. This way, the student experimented with the demands and techniques of searching and citing relevant information, and he was challenged and guided to analyze the results. In this case study, state of the art was built based on the following two crucial sources of information:

Systematic Review of Scientific Publications

For analysis of the related work, articles addressing automatic data extraction from invoices and receipts were studied. The Scopus platform was used as a data source. Scopus is one of the most complete databases in several areas and provides an advanced search that allows to configure search words in different fields such as titles, keywords, and full text, among others. It also allows the addition of logical operators: AND, OR, and NOT. This way was possible to access a significant number of scientific articles in computer science related to the theme of this project.

To collect the first sample of articles, the following keywords were defined for the Scopus search fields: "Extraction," "Recognition," "Invoice," "Invoice fields," "Algorithm," "Software," "Application," and "Program." The search was applied to the fields, title, keywords, and abstract: ("extraction" OR "recognition") AND ("Invoice*" OR "Invoice fields") AND (algorithm* OR software OR application OR program*).

Applying this query and the restriction to consider articles published after 2010 (inclusive) led to a total of seventy-one results.

After reading the title and abstract of each article, the articles that did not fit were eliminated, obtaining, in the end, a total of eleven articles.

With the full reading of the resulting eleven articles, they were further divided into two groups, mainly because the focus of some was not what was desired. In the end, of the eleven articles selected, four shared the same focus as the proposed project, and seven carried out only part of the proposal to be implemented. These seven articles that correspond only to a part of the implementation perform data extraction or comparisons between the algorithms used to do the extraction. That is, they only try to show various plausible algorithms, and afterward, the information is not allocated to any system.

In summary, we are left with four articles to analyze whose focus is equivalent to the application intended in this work.

Each of the four selected articles was analyzed and described considering the following criteria:

- Year of publication.
- Brief description of the objective of the work.
- What are the most common problems or scenarios intended to be solved by extracting data from invoices?
- What is the application's target audience (domestic, industrial, commercial)?
- What is the form of user interaction (web, mobile, web + mobile)?
- What is the detection algorithm (proprietary, how it learns, gives to various types of invoices, external API, Azure, among other considerations)?
- What is the result of the work (accuracy, types of invoices (several or just one), among other aspects)
- Main conclusions identified.

An analysis of the four resulting articles considering these criteria was then performed. This allowed all team members to clearly view related works and position the desired goals within those scopes.

Online APPs Overview

Also, the analysis of existing applications/systems whose purpose is identical to this work was carried out. The applications were found based on searches in the Google and Apple stores by applying filters such as keywords ("expense management", "expenses" and "financial control") or by selecting a particular category in which these applications fit (Ex. Finance). Here five applications were selected based on the following criteria: the platforms where it is possible to run these applications and the cost of acquiring them. Besides the five applications selected from the stores, three other applications were also considered available on the Internet but outside the stores. These applications were found by searching on the Google search engine using two strings: "capture data from invoice software" and "capture data from invoice."

All eight selected apps were analyzed and summarized. This, as done in the case of articles, allowed us to know better what products already exist for similar purposes and the most important features of related works.

Main Conclusions for the Design of the New Application

The four articles, the five applications selected from stores, and the three applications selected from additional searches allowed, at an early stage of the project, the identification of relevant functionalities, some of which were incorporated in the design and implementation of the Household Accounting project. In addition, it was possible to observe through the analysis of the article's different types of technologies and algorithms for invoice information recognition that could have been used to implement the new application and to understand better how necessary it is to have applications whose purpose to automate the reading of invoice information both at a business and personal level.

All this research work allowed the student, and the team, to have a better view of the state of the art. One of the main contributions of this project stage was to prepare the student for research and stimulate analysis and criticism skills. This allowed a clear and deep understanding of possible paths for the solution and opened a pathway for a better definition of the functional requirements. This led to a better definition of the functional and non-functional requirements defined after that by the product owner (explained below) also. In addition, it was essential to contribute to the team's cohesiveness from an early stage. The student showed interest in the subject, in knowing about it, and understood that the objectives were open and that everyone could contribute with improvement

proposals whenever this added more value to the product and aligned with the objectives defined by the company.

C. People/Team

In this case, the agile team was composed of 6 members, following the recommendations of Scrum [2]. Regarding the role of each one: 1 member of the company acted as product owner; 2 members of the company (with vast experience in terms of development using the adopted platforms) acted as coaches/development technical support; 1 member was the student that acted as a developer; 2 teachers acted as Scrum masters and, for some tasks, as coaches (involved in documentation, timeline, among others). In this process, the student, the central element of the approach, interacted with other people. Besides getting support for developing the project/product, he also gained experience in teamwork (soft skills), realizing the difficulties and aspects common in business projects of this type. The members' posture was demanding and methodical, continually adopting practices equal to what is done in the day-to-day business activity.

D. Process

Tools were adopted for this purpose to carry out the development process. Thus, Jira [15] was used to manage all stages so that details of the evolution of the project and its rhythm could be adequately monitored in an articulated manner. This choice requires everyone to follow good communication practices and compliance with activity logs and user stories in this case.

Figure 1 presents the timeline of one of the semesters.

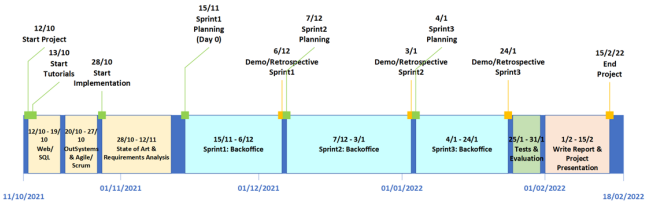


Figure 1. Timeline.

Scrum's artifacts [2] were all met, such as having a product backlog, sprint backlog, etc. In addition, there were daily meetings between the student and his mentors and checkpoints to clear any impediments to progress. The sprints were 2-4 weeks long, but every week there was a meeting (weekly meeting) between all the team members to review the progress of the work. There were sprints for development, but there were also periods when the goal was to learn how to develop or optimize the project; for example, how to integrate Azure [16] cognitive services into the product under development. In addition, the definition of the sprint periods was separate from the academic activity, which took place in parallel so that the student could also fulfill the academic requirements in his other subjects. Thus, there were different sprint periods as there were also different workloads.

E. Project

Since an agile approach was adopted, it followed the value [17]:

"Working software over comprehensive documentation."

In terms of requirements documentation and modeling, the requirements were documented using user stories, and in addition, we used wireframes and the Entity-Relationship (ER) model. The team did not follow an extensive and deeper documentation approach because the student knew it from previous work in other curricular units. However, taking advantage of this knowledge, the student also represented the use cases and the ER model for the final report.

The research work was demanding for the student and the other members involved. New challenges were posed that required research, pre-experimentation, and analysis. For example, to implement the synchronism between the mobile application and the backend, it was necessary to analyze several patterns and adjust them to the concrete objective of this new product. The same happened in relation to security aspects of the application or the use of Azure cognitive services. In other words, the fact that it is an application with ambitious goals also posed interesting challenges to all team members. The product owner defined and documented the initial requirements in the product backlog. The user stories' acceptance criteria were defined, which helped design the test cases and thus contributed to a robust application.

Although the student had general knowledge about Artificial Intelligence (AI), it required them to be prepared to take advantage of the resources provided by Azure in terms of parameterization and integration and in training to get the best performance. This is important because what was at stake in this challenge was to implement the functional requirements and user stories and obtain a final product with the highest possible accuracy in terms of automatic detection of fields of interest present in invoices.

Thus, the project involved research, development, software integration, application synchronization (Web and mobile), security, agile Scrum framework, teamwork, new tools (low-code platform, integration with cloud Azure, cognitive services, Jira, etc.). A detailed report of all phases and details of each aspect covered during the implementation process was also made. In the final stage of the project, acceptance tests were done to determine if the implemented features were useful and satisfied the users' needs.

This work covered many aspects of a software project, which could hardly be contemplated in a purely academic project. In addition to the technical-scientific coverage evidence, the agile methodology was chosen, and the fact that there was permanent communication between all its members was central. This leads us to verify in practice that one more value of the agile manifesto followed results in a successful path [17]:

"*Individuals and interactions over processes and tools*"

All these aspects mentioned above were considered, and the project includes documentation on user stories, database modeling, wireframes, and systems software architecture, among other valuable and necessary documentation.

Figure 3 shows the general architecture of the implemented system.

This work involved full-stack Web and mobile development using the OutSystems low-code platform [18].

This choice, teamwork, and adopting the agile framework (Scrum) allowed us to design from scratch and implement and test a complex and challenging software product during the standard school year.

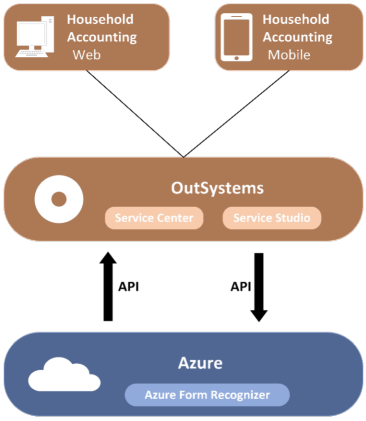


Figure 2. Systems Architecture.

F. Product

The recognition and automatic extraction of data from documents (invoices, receipts, etc.) are complex to implement and require various aspects to make it work successfully. With this project, we apply mechanisms to recognize and extract data from invoices and store, organize, and manage these data.

Using the OutSystems platform to develop the current project was also a requirement from the company. The company proposed this product idea to develop a Web and mobile application using the OutSystems low-code platform, allowing users to manage their expenses in a digital format, independently of the users receiving the documents digitally or on paper. One of the characteristics of this platform is the speed of development and the integration with other necessary tools for the implementation of the objectives of this work (e.g., integration with Azure services). It allows to build and deploy full-stack Web and mobile applications [18].

The product owner proposed the product backlog. However, in each sprint review meeting, there were adjustments to the user stories. A sprint retrospective was always carried out to keep the improvement process constant from sprint to the next sprint, fostering a continuous pace. This demonstrates to the student the importance of the third and fourth values of [17]:

"Customer collaboration over contract negotiation"

and

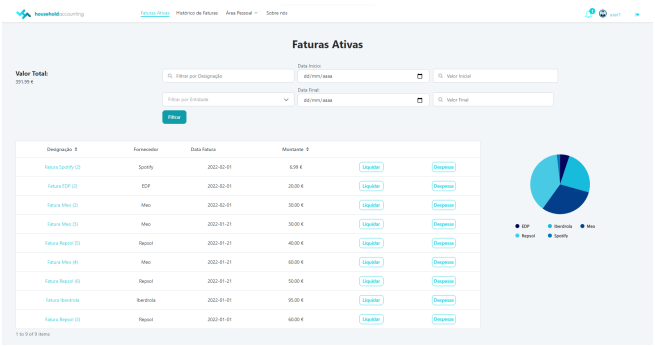
"Responding to change over following a plan"

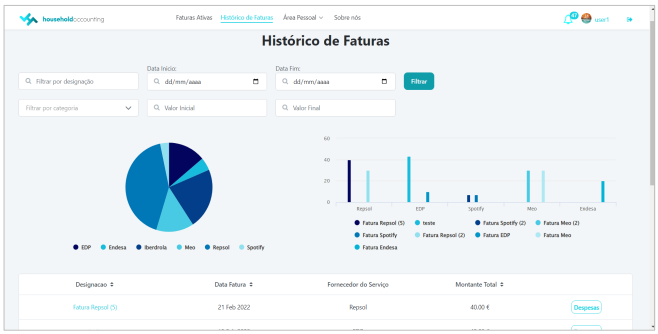
The final product was developed on time, and all goals were achieved. In other words, at the end of the project, the resulting product was an application (Web and mobile) that was developed in OutSystems with the integration of Azure cognitive services (Azure form recognizer [19]) that allows (among other functionalities) the user to:

- Register invoices automatically.
- Process invoices (recognize and extract) data from pdf or an image captured by a smartphone.
- See spending statistics of a specific type and period.

The mobile application was implemented to be used even when it is offline. Because of that, mechanisms to synchronize both applications (Web and mobile) were implemented.

Figures 3, 4, and 5 show the final layout of the Web portal User Interface (UI) and one of the mobile application's UI.





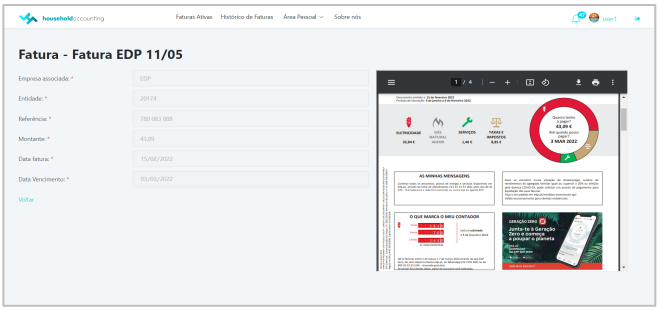


Figure 3. Examples of portal Web UIs (On top: active invoices, middle: Invoice's historical view, Bellow: Output of automatically processed invoice view).

Figure 3 (top) presents a dashboard with the active invoices (due to be paid). In Figure 3 (middle), we can see a dashboard to consult invoice's historic, with filters, charts presenting the collected data of different service providers (Telecommunications companies, electricity suppliers, etc.) and the list of stored invoices by designation, date, service provider, and monetary value. In Figure 4 (below), we can

see the result of an automatically processed invoice view, presenting the invoice's file and the automatically captured fields.

Users can consult their invoices that are due to be paid. They can choose to visualize it through a graph organized according to the value and entities of the invoices (Figure 4- left) or to visualize it in the form of a list (Figure 4 - right).

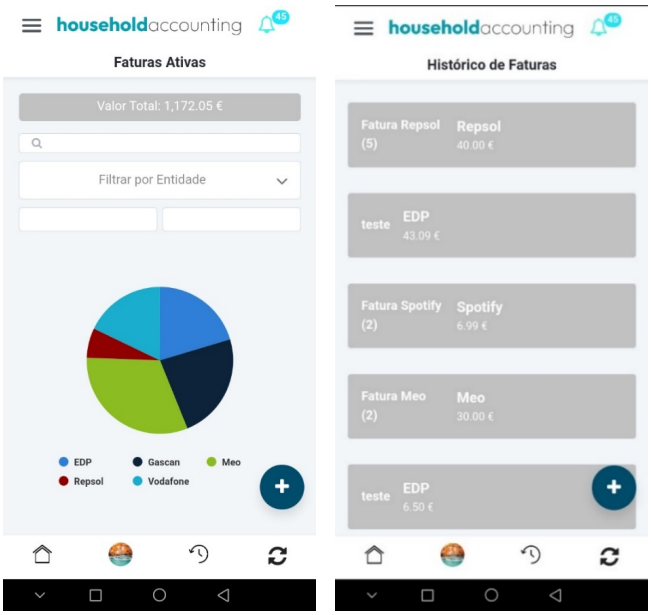


Figure 4. Examples of mobile app UIs (Left: active invoices- graph view Right: active invoices- list view).

In Figure 5, you can see the functionalities related to selecting the method to insert a new invoice (left) and a visualization of the user's expenses (right).

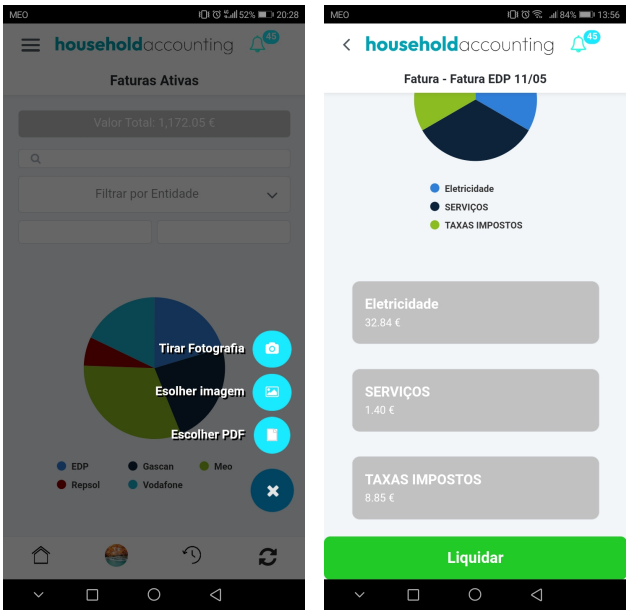


Figure 5. Examples of mobile app UIs (Left: select new invoice; Right: expenses).

Invoice templates can be configured in the administration portal for different service providers. The training and configuration of the recognition algorithms, using Azure cognitive services, can be configured through a dedicated Web portal.

The company's representatives validated the visual aspects (UI/User Experience (UX)), the synchronization between the Web and the mobile applications, and security issues. The performance results obtained with the recognition of invoices were also analyzed and improved.

In technical terms, several aspects have been studied from scratch and implemented for this software application, demonstrating a real and active learning implementation. Some of the aspects that evidence these practices are:

Security

The constant demand to ensure the security of information and infrastructures is recurrent and essential. In this project, some measures were considered to ensure a higher level of se curity for users and the information associated with using the mobile application.

The security mechanisms implemented in the mobile application follow the Top 10 Mobile Risks identified by the OWASP organization [20] in 2016. The ten risks identified by the organization were as follows: M1: Improper Platform Usage; M2: Insecure Data Storage; M3: Insecure Communication; M4: Insecure Authentication; M5: Insufficient Cryptography; M6: Insecure Authorization; M7: Poor Code Quality; M8: Code Tampering; M9: Reverse Engineering; M10: Extraneous Functionality.

Local Data

Creating a local database becomes mandatory once it is decided to allow the user to use the application without an Internet connection (offline). As such, using the application offline requires that some of the user data and associated data be stored on the device itself. Using the application offline does not allow using all the features that the concept promises; however, it ensures most of them, such as viewing active expenses.

The information stored in this application includes data from the invoices and their respective expenses (information from the logged-in user's invoices and the expenses associated with those invoices), from the support messages (associated with the logged-in user), from the notifications (associated with the logged-in user), from the logged-in user, from the proposals of new types of invoices (information from the proposals made by the logged-in user and from the user who generated the proposals) and data related to the application itself (properties for running the application).

The information stored in local storage at the login stage usually does not correspond to all the data stored in the server database. The amount of data stored in the server database can be huge, and it is necessary to filter the amount of data to be stored on the end device, namely by how recent the data is.

The structure of the proposed local database was based on non-relational modeling, i.e., NoSQL modeling. NoSQL modeling is used to process large volumes of unstructured

and ever-changing data. This type of modeling is very efficient in data processing and fast in querying information [21]. NoSQL modeling on mobile devices is highly important because devices do not have the same computing power as a server. This type of constraint has repercussions on the use of the mobile application, such as an increase in the application's response time to the user. The relationships between tables require more computational power and, consequently, more time. That said, the structure of each table stores specific information and associated information, possibly relevant when accessing the primary information.

Although two structures are used to store the authenticated user information, the server database is the main structure that guarantees data fidelity.

Synchronization

Synchronism becomes necessary when the mobile application allows users to use features without (offline) and with an Internet connection.

For the user to use the application without being connected to the Internet, saving some user information and some associated data is necessary. Given this, the user will use the application the usual way regardless of whether he has an Internet connection, causing the data/records to change.

In offline mode, it is necessary to ensure that the data/records that have changed in the device's local memory while using the mobile application without an internet connection will be updated in the server database when reconnecting to the Internet. In short, synchronization is based on ensuring that the data in the server database and the data in local storage are always up to date with the latest version of the data that has been manipulated and ensuring that the navigation is automated so that it is not dependent on a constant Internet connection. The concept and methods inherent to synchronization can be quite complex to understand and implement, and synchronization is achieved through implementing platform-specific patterns called Offline Data Sync Patterns [22]. The data synchronization patterns used in the mobile application were the following:

- Read-Only Data [23];
- Read/Write Data Last Write Wins [24].

The Read-Only Data pattern is essential for storing auxiliary information, such as filling dropdowns.

The use of the pattern Read-Write Data Last Write Wins is important, in turn, for synchronizing the changed information in the server database and the local database. This implementation makes it possible to use the mobile application without the user being connected to the Internet.

Data synchronization is performed when a given user logs into the mobile application (the essential data is loaded at this stage), on-demand (as the application is used continuously), and when the user uses the *PullToRefresh* feature implemented on almost every screen.

UI/UX

The development of mobile applications also requires a special concern with the design of interfaces and the experience that affects the end user. In this project, there was a constant concern with the interface design and providing the users with the best usability experience. Some of the details (easily visible using the mobile application) that were considered were the following: Size of the buttons; Button position; Color of the buttons; Representative icons; Contrast between colors; Always keeping the user informed. However, some programming logics are implemented to provide the best user experience; for example, the *OnScrollEnding* mechanism is used to load more data on pages that list information. The algorithm implemented loads only some information at a time to load the page as fast as possible and does not immediately make reading the data exhausting for the user. As such, the data is loaded in blocks of fixed size according to the user's wishes.

Performance

The performance of mobile applications has, over time, increased with technological advancement. However, this advancement does not evolve in only one direction. While hardware evolves, the amount of data that users manipulate and generate has also been increasing, in addition to the resources used by the applications.

The application solutions were implemented throughout the development to provide a better user experience. Some of the solutions identified and implemented to improve the performance of the application were the following:

- Restrictions on the amount of data to be stored in the local database.
- Restrictions on the amount of data to display to the user when loading certain screens.
- Image compression.

The amount of data associated with a given user can be immense, and one must restrict the amount of data to be stored in the local database without compromising the normal operation of the application without the device being connected to the Internet. A practical example implemented in the Household Accounting application corresponds to the information on outstanding invoices and the respective expenses stored in the local database.

The main page of the mobile application lists the outstanding invoices of the authenticated user; however, the invoices are not displayed all at once so as not to cause significant delay when building the page. That said, fixed-sized blocks are loaded at a time (depending on the user's willingness to scroll the page). First, the page is filled with data in local storage, and if all the data in local storage has already been loaded, it is checked to see if there are any new records in the server database that do not exist in the local database. If there are new records, they are shown on the screen.

The main functionality implemented in the mobile application is the automatic reading of relevant information from an invoice. Reading this information from invoices can be done by uploading PDF files, uploading gallery images taken previously from invoices in physical format, and taking a picture of the pages of a physical invoice at the time. Using this functionality in the last-mentioned format caused some inconvenience regarding the size of the files being generated,

which would have to be handled by the application and the resource. In the first prototypes, the usage of the application without performing any "treatment" on the generated files led to the closing of the application after a waiting period (3-4 minutes) for the application's response to the reading of the invoice by the resource. The application generated files with about 8 MB (for each page of an invoice). This made the application unable to handle such a large amount of data and sending data to the Azure resource with a size exceeding 15 MB (in the case of an invoice consisting of two pages) because analyzing the invoice information takes time, taking up to 4 minutes and/or even closing the application suddenly. Considering this, it was necessary to implement some compression mechanisms to solve this problem. In the first instance, a comparison was made between two methods provided by two different OutSystems components to proceed with the compression of the generated files. To do this, several factors were considered, such as the time of performance, the size of the files obtained because of the methods, and the image quality lost with each one. The comparison was made between the following methods provided by the following OutSystems components:

- *Image_Utils_Reactive*.
- *ImageCompress*.

To select the most appropriate method, some initial experiments were done. Table 1 presents the results of the experiments using these two methods applied in the same file.

TABLE I. OUTSYSTEMS COMPONENTS COMPRESSION.

Image_Utils_Reactive Before (original)		Image_Utils_Reactive After (60% quality)	
Size	8 MB	Size	629 KB
Width	2268	Width	2268
Height	3024	Height	3024

ImageCompress Before (original)		ImageCompress After (60% quality)	
Size	8 MB	Size	739 KB
Width	2268	Width	2268
Height	3024	Height	3024

Initial experiments start by compressing the captured image to 60% of its initial quality. These experiments showed that the images compressed through the *Image_Utils_Reactive* component are smaller than those compressed by the *ImageCompress* component (Table 1 up-right/down-right). It was also verified that the use of the component did not result in such a long delay for the final insertion of a new invoice.

However, although the component generated a smaller file size, it was noticed that the algorithm's accuracy had been compromised when the file was compressed too much. Then, more experiments with distinct compression rates (considering 70% and 80% of the image's original quality) were performed with the *Image_Utils_Reactive* component. These experiments made it possible to realize that file compression below 70% is the threshold for correctly understanding invoice information.

Integration with Azure Services

Azure Form Recognizer [25] is an artificial intelligence (AI) service provided in the cloud that uses AI models to extract information from documents such as text and tables from our documents. In this case, the documents are invoices, and extracting the desired information from the invoices was possible.

The configuration of the template to be used in this system involved the use of a tool called Form OCR Testing Tool [26]; however, the template's configuration required the existence of some previously obtained keys because of the configuration on the Azure side. Invoices uploaded to Azure containers (in various formats) appear in the editor automatically, ready to be configured. It is then necessary to run a function on each document to highlight which fields are possible to select to be identified by the template.

Afterward, it is necessary to create tags (as you can see on the right side of Figure 6) for each piece of information you want to extract from the invoices. The tags considered in this implementation were 14.

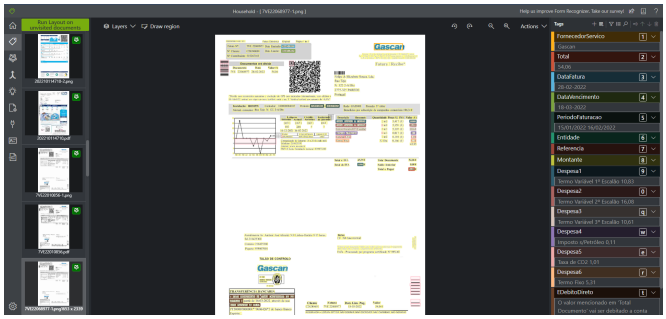


Figure 6. OCR Testing Tool GUI.

Finally, the model was trained based on the configurations performed previously. It required the configuration of at least five invoices to be able to be performed, and obviously, the more invoices inserted and configured, the better the training results.

The results of the training vary, considering the number of files inserted and configured on the platform, the concern with the tags that are created and how general they can be to respond to all types of invoices, and, finally, the insertion of files of various types and obtained in various ways. PDF files and PNG files were used and converted from invoices in PDF format. PNG images obtained from photographs of paper invoices were also inserted.

During the study stage, it was noticed that the resource's accuracy in analyzing invoice photographs is lower than in analyzing PDF invoices. For example, when using a PDF file consisting of photographs taken from the camera of an end device, information near the edges of the pages was identified with flaws. This pattern is easily justified by the fact that the focus of the devices' cameras is by default on the center, so the quality of the remaining areas of the photos will be lower.

Integration with *OneSignal*

Another deployed issue was using *OneSignal* [27], which provides push notifications for mobile devices and web,

messages within the application, SMS, and email. The use of these services is possible through the free consumption of the API by REST. Despite being the free version, the version of *OneSignal* used in our application guarantees the data protection imposed by the General Data Protection Regulation (GDPR). However, they use some statistical values for personal marketing purposes, among others. This tool was useful in integrating the native notifications of users' end devices with our application, providing further rigor to the mobile application.

Above all, it was another tool to integrate into our application that, consequently, led to learning how it works and how it is possible to integrate it into the most diverse systems. Furthermore, it was also important to demonstrate how the functionality works and observe the intended goal with the integration of *OneSignal*.

These are some of the several (others exist like database modeling and implementation or learning new tools (ex. balsamic)) aspects involved in this project's preparation, design, and implementation. This demonstrates the wide range of software aspects and tools involved throughout the project and at various levels (in terms of technical knowledge and soft skills). The project's success was due to the student's commitment and dedication and to the entire collaborative environment that was present from the beginning. We believe it was clearly a success story that we intend to replicate in subsequent editions and with other students.

IV. RESULTS AND LESSONS LEARNED

According to the opinion of all those involved, the result of the project was very positive. In addition to completing the entire system within the planned period of 2 semesters, a high-quality software product was developed (all requirements implemented and good acceptance from potential external end-users). In other words, in addition to providing all the intended features identified throughout the project, the software developed also performs with good performance results. After several tests with invoices from various service providers, the performance was excellent in all cases. As in any of these cases, the better organized the information on the invoice is (input), the easier it will be to train the system and, obviously, the better the accuracy of the data obtained (output). In the tests carried out, in most cases, all data was recognized automatically from the original pdf invoices received by email from the service providers (e.g., a gas company, an energy company, or a telecommunications provider). A lower accuracy rate was achieved if the invoices were digitized using the smartphone camera (even so, in the performed experiments, at least 46.5% of the fields were well recognized, and the user manually entered the remaining fields). After having the first version of the system available (Web and mobile), several potential users were asked to install and use the application and to respond to a survey. The survey included 14 questions. Twelve questions were answered using the Likert Scale (1–5), and one question asked for a numerical answer. The other question was an optional free-response question where respondents could include any information. The results obtained at this stage serve three crucial goals: 1) to provide a better insight into the platform, which may identify novel issues/problems to consider; 2) to obtain initial feedback on potential users' acceptance and perception of the platform's key features and 3) to evaluate the usefulness of the proposed system. Twelve users completed the survey. To exclude the outliers, the survey with the best evaluation and the survey with the worst evaluation were excluded. This resulted in 10 valid answered questionnaires.

The analysis of the responses shows that 90% of respondents rated the application as useful or very useful, and 80% rated it as easy or very easy to use. As shown in Figure 7, all the respondents were satisfied or very satisfied with the automatic reading of invoice information.

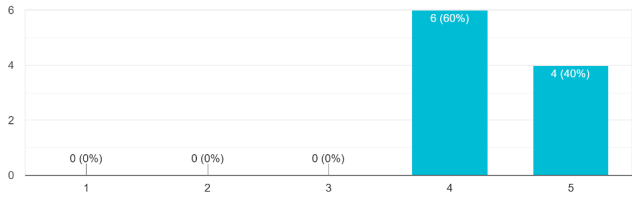


Figure 7. Users' satisfaction related to automatic reading of invoice information.

The functionalities that allow the import of invoices from PDF files or photos are among the most relevant in the application. Regarding these features, user opinions were very positive (see Figure 8). Importing invoices in PDF format was considered very important by 80% of respondents (Figure 8).

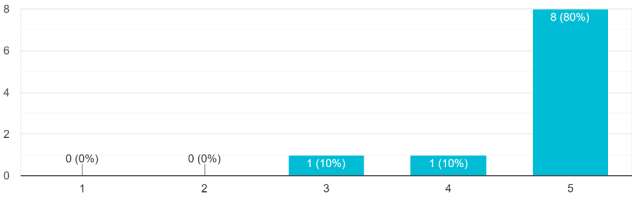


Figure 8. Importance of the "importing invoices in pdf format" feature.

Users recognize that the feature import of invoices from a photo was important or very important by 60% (Figure 9). However, despite the recognition of the importance of this feature, user opinions are more divided. Only 40% of them recognize this functionality as important or very important.

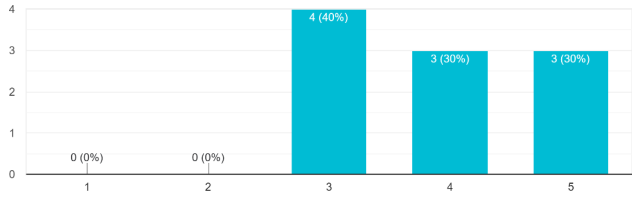


Figure 9. Importance of the "importing invoices from photo" feature.

How data is identified (extracted from the invoice) and presented to users was also to the liking of respondents, as can be seen in Figure 10. All respondents rated how the data are presented as good or very good.

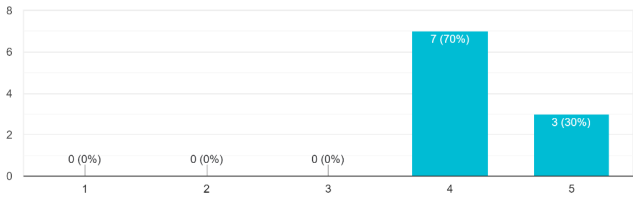


Figure 10. How the data extracted from the invoice is presented.

In the open-answer question, it was possible to obtain some feedback on usability improvements and the reporting of some bugs.

In terms of lessons learned, this approach requires a dedication of at least one hour/week (average) from the teachers and the company's members. In the case of the mentor, this period was longer due to all the daily meetings. The dedication paid off because the result (resulting product, preparation of the student (technical and non-technical skills)) was very positive. The fact that everyone was engaged in developing a comprehensive project that involved all stages and components of the proposed architecture was challenging, motivating, and clearly beneficial for all parties, including teachers, students, and staff involved from the partner company.

This approach has been successfully carried out in this case study and it was also implemented in previous editions with other students with different characteristics [28]. In both scenarios very positive results were achieved which suggest that this approach can contribute to overcome the gap academia-industry, mentioned in the introduction of this article. This approach fosters students to develop state-of-the-art technical and non-technical skills. However, these case studies also showed that this is a demanding approach that depends on the availability of all participants: teachers, industry partners and students. Scaling up the application of this methodology to a large number of students will be a challenge as it will require availability and a lot of work on the part of teachers and professionals from the companies involved. In this sense, to ensure the success of this methodology, an important aspect is that all parties (industry and academia) should be committed to apply this approach and to guarantee the necessary time availability. Also, students enrolled in this approach must be aware that they must maintain a constant pace and commitment with the established goals.

V. Conclusions

In this work, we presented a case study that shares an agile approach to preparing students for the job market regarding SE practices. The main goal of this approach was to prepare students' performance and technical skills for the software industry's new demands and thus contribute to reducing the gap between SE education and practice in the software industry. We followed an approach that promotes active and collaborative learning to achieve this goal. The case study presented illustrates the work methodology, the approach, details of the project design and implementation, the results achieved, including the resulting product, some lessons learned, and some guidelines for the future. And the resulting product.

The student was involved in a distributed team with teachers and IT professionals from a software house to develop a product that demanded full-stack development and agile best practices. These industry-academia partnerships help students become better and more quickly prepared to work in high-performing teams. They raise students' employment opportunities by preparing them in cutting-edge fields and improving their soft skills to perform better in software development teams. These partnerships are also advantageous for the other partners involved. Hiring qualified human resources is good for the companies, as well as for the participating higher education institutions (contributes to improving their employability rate).

In this case study, an Agile methodology was followed. It allows closer and more frequent monitoring of all members of the team and therefore has advantages for student learning. In situations where companies use other methodologies or development tools, the involvement of companies in the student's teaching/learning process will always benefit the student. However, to assess these benefits or to compare the benefits when different methodologies are followed, further studies will be needed.

Acknowledgment

We thank the members of the company Do iT Lean, who were also involved in this case study providing technical support.

References

[1] J. Metrôlho, F. Ribeiro, and P. Graça, “Prepare Students for Software Industry. A case study on an agile full stack project,” in *Seventeenth International Conference on Software Engineering Advances*, 2022, pp. 75–80.

[2] K. Schwaber and J. Sutherland, “The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.,” 2016. https://www.scrum.org (accessed Jul. 20, 2022).

[3] R. Brungel, J. Ruckert, and C. M. Friedrich, “Project-Based Learning in a Machine Learning Course with Differentiated Industrial Projects for Various Computer Science Master Programs,” in *2020 IEEE 32nd Conference on Software Engineering Education and Training, CSEE and T 2020*, 2020, pp. 50–54, doi: 10.1109/CSEET49119.2020.9206229.

[4] L. Gren, “A Flipped Classroom Approach to Teaching Empirical Software Engineering,” *IEEE Trans. Educ.*, vol. 63, no. 3, pp. 155–163, 2020, doi: 10.1109/TE.2019.2960264.

[5] P. Rodrigues, M. Souza, and E. Figueiredo, “Games and gamification in software engineering education: A survey with educators,” in *2018 IEEE Frontiers in Education Conference*, 2018, vol. 2018-Octob, pp. 1–9, doi: 10.1109/FIE.2018.8658524.

[6] R. Malhotra, M. Massoudi, and R. Jindal, “An innovative

approach: Coupling project-based learning and game-based learning approach in teaching software engineering course," in *Proceedings of 2020 IEEE International Conference on Technology, Engineering, Management for Societal Impact Using Marketing, Entrepreneurship and Talent, TEMSMET 2020*, 2020, pp. 1–5, doi: 10.1109/TEMSMET51618.2020.9557522.

[7] W. E. Wong, "Industry Involvement in an Undergraduate Software Engineering Project Course: Everybody Wins," in *120th ASEE Annual Conference and Exposition*, 2013, pp. 23.742.1-23.742.12, doi: 10.18260/1-2--19756.

[8] E. Tuzun, H. Erdogmus, and I. G. Ozbilgin, "Are Computer Science and Engineering Graduates Ready for the Software Industry? Experiences from an Industrial Student Training Program," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2018, pp. 68–77.

[9] A. Heberle, R. Neumann, I. Stengel, and S. Regier, "Teaching agile principles and software engineering concepts through real-life projects," in *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2018, pp. 1723–1728, doi: 10.1109/EDUCON.2018.8363442.

[10] G. Wedemann, "Scrum as a Method of Teaching Software Architecture," in *Proceedings of the 3rd European Conference of Software Engineering Education*, 2018, pp. 108–112, doi: 10.1145/3209087.3209096.

[11] I. Bosnić, F. Ciccozzi, I. Čavrak, E. Di Nitto, J. Feljan, and R. Mirandola, "Introducing SCRUM into a Distributed Software Development Course," in *Proceedings of the 2015 European Conference on Software Architecture Workshops*, 2015, pp. 1–8, doi: 10.1145/2797433.2797469.

[12] J. J. Chen and M. M. Wu, "Integrating extreme programming with software engineering education," in *38th International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2015, pp. 577–582, doi: 10.1109/MIPRO.2015.7160338.

[13] B. S. Akpolat and W. Slany, "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification," in *27th Conference on Software Engineering Education and Training*, 2014, pp. 149–153, doi: 10.1109/CSEET.2014.6816792.

[14] S. Al-Ratrout, "Impact of using Agile Methods in Software Engineering Education: A Case Study," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 1986–1991, doi: 10.1109/CoDIT.2019.8820377.

[15] ATLASSIAN, "Jira Software." https://www.atlassian.com/br/software/jira (accessed Oct. 13, 2022).

[16] Microsoft Corporation, "AZURE. INVENT WITH PURPOSE. Learn, connect, and explore." https://azure.microsoft.com/en-us/ (accessed Oct. 13, 2022).

[17] "Manifesto for Agile Software Development," 2001. https://agilemanifesto.org (accessed Jul. 20, 2022).

[18] OutSystems, "OutSystems Developers: Develop more. Ship more. Get more done." https://www.outsystems.com/developers/ (accessed Jul. 26, 2022).

[19] Microsoft Corporation, "Azure Form Recognizer." https://azure.microsoft.com/en-us/services/form-recognizer (accessed Jul. 20, 2022).

[20] OWASP, "Top 10 Mobile Risks - Final List 2016," 2016. https://owasp.org/www-project-mobile-top-10/2016-risks (accessed Mar. 21, 2023).

[21] Microsoft Corporation, "Base de Dados NoSQL – O que é NoSQL?" https://azure.microsoft.com/pt-pt/overview/nosql-database/ (accessed Mar. 21, 2023).

[22] OutSystems, "Offline Data Sync Patterns," 2023. https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Offline/Offline_Data_Sync_Patterns (accessed Mar. 20, 2023).

[23] OutSystems, "Read-Only Data," 2023. https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Offline/Offline_Data_Sync_Patterns/Read-Only_Data (accessed Mar. 20, 2023).

[24] OutSystems, "Read/Write Data Last Write Wins," 2023. https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Offline/Offline_Data_Sync_Patterns/Read/Write_Data_Last_Write_Wins (accessed Mar. 20, 2023).

[25] OutSystems, "Get started with the Form Recognizer Sample Labeling tool," 2023. https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/quickstarts/try-sample-label-tool (accessed Mar. 20, 2023).

[26] Microsoft, "Microsoft, "Editor - Form OCR Testing Tool." https://fott-2-1.azurewebsites.net/ (accessed Jun. 22, 2022).

[27] OneSignal, "OneSignal." https://onesignal.com (accessed Jun. 13, 2022).

[28] J. Metrôlho, F. Ribeiro, P. Graça, A. Mourato, D. Figueiredo, and H. Vilarinho, "Aligning Software Engineering Teaching Strategies and Practices with Industrial Needs," *Computation*, vol. 10, no. 8, 2022, doi: 10.3390/computation10080129.

Methodological Choices in Machine Learning Applications

Kendall E. Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: kendall.nygard@ndsu.edu

Mostofa Ahsan, Aakanksha Rastogi, Rashmi Satyal
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: {mostofa.ahsan, aakanksha.rastogi, rashmi.satyal}@ndsu.edu

***Abstract*—Machine learning is a subset of artificial intelligence in which a machine has an ability to learn and employ complex algorithms to impersonate human behavior. Development of a machine learning model involves careful preparation and management of data and selection and features to produce meaningful results. The data issues are often challenging due to availability, characteristics, properties, categorization, and balance. We report on relevant literature, case studies and experiments surrounding the data issues. We describe alternative machine learning methodologies and emphasize supervised learning, including treatment of experimental procedures. Procedures and challenges in the collection, quantity, distribution, quality, sampling, of and relevancy of data are included. Applications of machine learning models are presented, including classification models for self-driving cars. These models introduce anti-autonomy trust modeling. We also describe intrusion detection models that can detect malicious activity in computing systems. These applications also provide insight into overfitting and underfitting training data. Feature engineering and feature selection issues are presented, including approaches to identifying, combining, and eliminating attributes and features to determine which are needed and their significance. Approaches for treating class imbalances in data management are discussed. Comparisons among categorical encoding techniques are presented. The work provides perspective and insight into resolving multiple issues that must be addressed in utilizing machine learning models in practice.**

Keywords- Machine Learning; Data Management; Feature Engineering; Feature Selection; Self-Driving Cars; Intrusion Detection.

I. INTRODUCTION

Machine Learning (ML) is a rapidly emerging area of artificial intelligence. Many types of applications have been successfully developed and new successes are regularly reported. The famous Turing award winner Jim Gray referred to data science as a "fourth paradigm," taking a rightful place among empirical, theoretical, and computational sciences [2].

Often viewed as interdisciplinary, data science involves mathematics, statistics, and computer science as well as other related areas. In many applications, the availability of large and relevant datasets, and the methods of data science provide the lifeblood of machine learning problem-solving approaches. Analyses and decision support in nearly every area of human endeavor today are related to machine learning.

The example machine learning studies that we describe are in the areas of self-driving cars and intrusion detection [1] [3][34][37][38]. Self-driving cars are a real-world example of a system that requires machine learning, since they involve complex computations, algorithms, computing systems, mechanics, and behavioral aspects that endanger human life if automated decisions or controls go awry. Automated cars have features such as remote engine start, advanced information systems, moving object detection (MOD), lane change assist, anti-lock braking system (ABS), to name just a few. These systems can create vulnerabilities to cyber-attacks from things like bugs introduced in their core software code, remote access to the onboard diagnostic system (OBD) of the vehicle, or controller area network (CAN) bus [77].

For a self-driving car study focused on classification, we utilized available multi-attribute data about specific collisions. The data contained many features and attributes of the vehicle itself, the damage incurred, roadway conditions, etc. The objective of the study was to build a classification model that could translate the detailed data into collision predictions and to drive an anti-autonomy trust model. There were several important and difficult choices made related to scale and balance within the available dataset, and in feature engineering. A linear sequential supervised machine learning model was employed.

The intrusion detection study used supervised learning techniques to build a model for identifying outside threats initiated by malicious actors who wish to breach or compromise a system. Among other datasets, the study examined the famous dataset that originated in the KDD (Knowledge Discovery and Data Mining) competition and was later modified to form the now publicly available NSL (Network Security Laboratory) KDD dataset [7][8].

Data management and feature engineering are important steps in the cybersecurity domain to prepare data for machine learning algorithms and build effective models for detecting security threats and anomalies. Data analysis involves exploring, visualizing, and understanding the data to identify patterns, trends, and anomalies [48]. This may include statistical analysis, correlation analysis, time series analysis, and other techniques to gain insights into the data. In the cybersecurity domain, data analysis may also involve extracting features that are relevant to the security domain, such as network traffic flow, packet size and content, system

logs, user behavior, and other indicators of security threats [49]. Feature engineering is the process of selecting, transforming, and creating features that are relevant and informative for the machine learning model. This may involve selecting features that are correlated with the target variable, transforming features to make them more informative or meaningful, and to create new features from existing ones [50]. Feature engineering in the cybersecurity domain may also involve domain-specific knowledge and expertise to select and create features that capture the specific characteristics of security threats and anomalies [51][52]. Feature engineering techniques may also involve dimensionality reduction, feature scaling, and other preprocessing techniques to improve the performance of the machine learning model. All these methods have their strengths and weaknesses, and their effectiveness may depend on the specific dataset and classification problem at hand. In the cybersecurity domain, they can be useful to improve the performance of machine learning models that aim to detect and classify security threats, anomalies, or attacks [53]. We describe some of the data analysis and feature engineering techniques used in cybersecurity in the data management section of this paper.

The rest of the paper is structured as follows. In Section II, we describe supervised machine learning with illustrations of the flow of a machine learning model and data splits for cross validation. In Section III, we present a self-driving cars example illustrating an implementation of a linear sequential supervised learning artificial neural network model utilizing multiple pre-processed complex attributes. In Section IV, we present the intrusion detection example, explaining how a machine learning model can be tuned to predict and identify attacks. In Section V, we describe data management and feature engineering issues that are ubiquitous in machine learning practice. This section also includes several categorical encoding techniques for preprocessing data for a machine learning algorithm. Finally, we conclude our work in Section VI. An abbreviated version of the work is available in [1].

II. MACHINE LEARNING EXPERIMENTS

Machine learning methods are of four distinct types. Supervised learning models are trained using datasets with known labels, then used to make predictions on new data. Unsupervised learning models work with unlabeled data and seek clusters or patterns in the data. Semi-supervised learning is a hybrid approach that uses both unsupervised and supervised learning techniques. Reinforcement learning uses no labeled data, but instead is based upon evaluation of rewards or punishments of behaviors. We have conducted extensive experiments using supervised learning in applications concerning circumstances under which collisions occur in self-driving cars and in detecting intrusions into computer platforms.

Supervised machine learning (SML) methods are very effective in addressing classification problems. When applied to classification tasks, a SML method has a set of available data for which their correct classifications are known. Such a data set can be represented as shown below.

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_n, y_n)$$

Here, the $\mathbf{x}_i$ shown in bold are vectors that capture a data instance or situation, and the corresponding y_i values are discrete labels for the available classifications.

The initial task in a supervised learning experiment is to computationally train the machine learning model to accept the known data instances as input and to produce the correct target as output. Many types of SML methods can be used in training, including decision trees, neural networks, support vector machines, and logistic regressions [78]. For some methods, training can be a computationally intensive process. Once trained, the model is available to accept new data instances and predict their target classification. The model is successful if it has high values of performance measures such as percentage of accuracy in correctly classifying the new data instances, called the ability to generalize. There are multiple issues surrounding the characteristics of the available data, the classes into which they fall, their attributes and features, and the learning models charged with producing the predictions. Concerning baseball, Coach Yogi Berra famously said, “It's tough to make predictions, especially about the future.” This aphorism is equally true in machine learning [46].

Figure 1 illustrates the general flow of a machine learning technique. Several tasks are included. The overall task of the DEVELOP phase shown at the top is to produce a Final Model that is fully specified, trained and feeds into the PREDICT phase shown below the dotted line, where it is available for generalization use on new data. Starting from the top, the data is shown as partitioned into splits for Training, Validation and Test. The full data is divided into the Training Split and the Test Split. A good way to perform training is to withhold a portion of the data while training is done. It is viewed as a mistake to train and test a machine learning model on the same data. So, doing that would result in the model memorizing all the data/target pairs, resulting in the model perfectly knowing all of the answers, leaving no ability to generalize. The result is known as *overfitting*. For validation purposes, the Training Split is typically divided into pieces called folds. Called k-fold cross validation, Figure 2 illustrates basic logic for splitting the data. In this example, k=5 so there are five equal parts. This corresponds to the Validation Split and Model Tuning blocks in Figure 1. In Figure 2, shown in bold italics on the diagonal, there is a designated fold in each row that is specified for testing, with the other four used for training. The key idea is to find the best set of meta-level parameters for a model being developed. All major machine learning models have parameters. For example, an ML that utilizes an Artificial Neural Network (ANN) in some way, will be parameterized with settings like Learning Rate (governs weight adjustments), topology (number of hidden layers, nodes

within layers, and interconnectedness), and activation functions.

In cross validation, when a model is trained on the folds, a performance metric, such as classification accuracy, can be calculated on the testing fold. After all the fold splits are evaluated in this way, an average is calculated, which yields a score for the parameter settings. Various optimization methods can be employed to explore the parameter space in a quest to identify the best settings. Viewed more generally, the Model Tuning block can also be viewed as exploring various types of models in a quest to not only optimize the use of one type of model, but to also choose among competing models.

In multiple places of the ML process, there is a need to evaluate the quality of the predictions using a metric. The empirical accuracy of a method is simply the percentage of the predictions made that are correct. Other metrics are available. More details are provided later in this paper.

Raw data is rarely available in a form that is suitable for direct use by an ML model. Pre-processing of data is typically necessary to deal with such things as null or missing values, outliers, transforming or reconciling numeric and categorical values, rescaling, and standardizing. We expand on the data-centric issues, for the example applications reported in this article.

Feature Engineering also appears in Figure 1. Features are those characteristics that are present in the data that are potentially useful in predicting a target outcome.

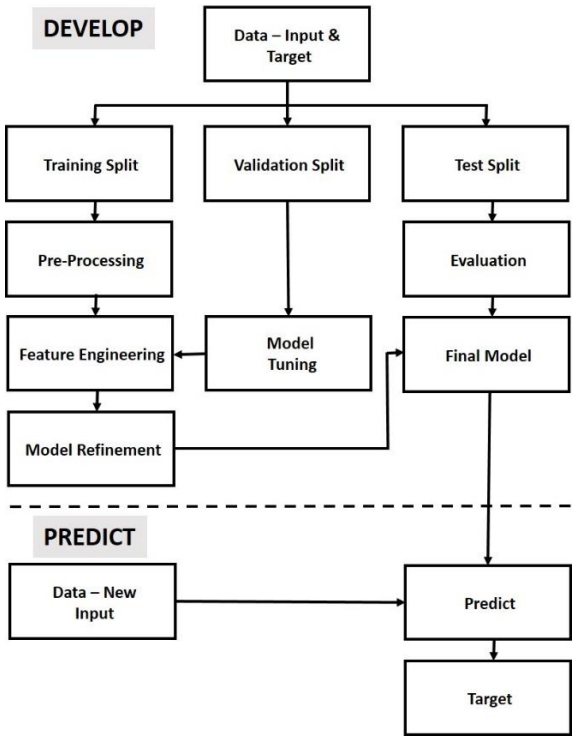


Figure 1. Flow of a Machine Learning Model [3].

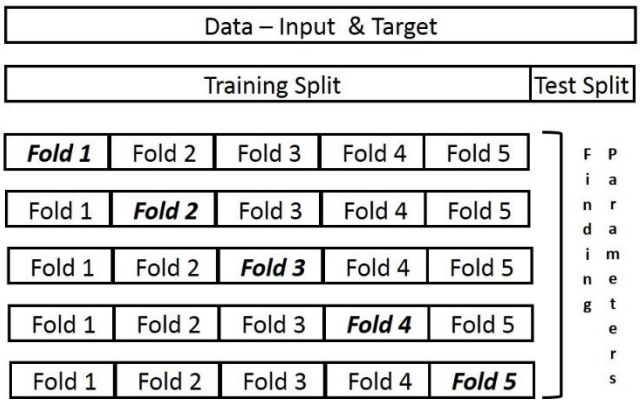


Figure 2. Data Splits for Cross Validation.

It is often effective in ML to modify or combine features in some way to produce a new feature that can improve the prediction accuracy of the method. Called Feature Engineering, the operations that can be carried out include things like mathematically transforming a single feature or applying a functional calculation on multiple features. Feature Selection refers to reducing the number of features employed by the model while retaining acceptable results. Reducing the features needed can ease the data collection task and reduce the computational load of running the model. Feature Selection typically follows Feature Engineering. We provide details related to the examples discussed in this paper.

Unsupervised Learning is different than supervised learning in many ways. Some of the most known algorithms are, k-means clustering, hierarchical clustering, principal component analysis, and a priori algorithm [45]. The need for unsupervised models is increasing in the cybersecurity domain since attacks are being modified every day [47].

We have discussed multiple machine learning techniques in this section. The primary concern is to make proper choice of methods to optimize the solution of a problem. We discuss the criteria we need to adopt to address a machine leaning problem in the following sections.

III. SELF-DRIVING CARS APPLICATIONS

A self-driving car underway must adhere to laws and rules of the road, like adhering to speed limits, stopping at red traffic lights, turning from an appropriate lane, etc. In addition, the vehicle must carry out control actions in response to sensor and communication information that provides information regarding things like conditions of the roadway (such as snow, rain, deteriorated pavement); construction zones; presence of bicycles, pedestrians, other cars, or obstacles; or visibility issues like glare, fog, snow, rain, darkness, or any type of impaired lighting.

When addressing how a self-driving car can be trained to carry out an appropriate action under circumstances that it encounters, we draw an analogy with how positive

reinforcement can work efficiently when a person or animal learns a new skill or behavior. This vehicle training need lends itself naturally to Reinforcement Learning (RL), a powerful machine learning technique that rewards good behaviors, and as needed, punishes bad behaviors. A RL training method proceeds iteratively and is illustrated in Figure 3 [76]. While undergoing learning in a simulated environment, the vehicle is an agent that carries out actions, receives their consequences in the form of positive or negative feedback, and adjusts their model of actions accordingly. When complete, experience gained in this way can maximize rewards. With acceleration, deceleration and steering as the primary actions, a self-driving car (agent) aims at maximizing short-term rewards (like safe driving) and long-term rewards (like fast arrival at the destination) using the RL approach.

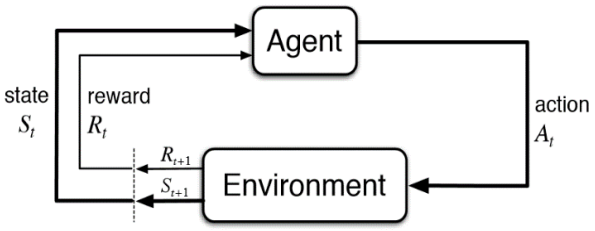


Figure 3. Reinforcement Learning illustration.

We carried out extensive experiments using supervised learning for analyses of collisions that occur with self-driving cars. Official collision reports basically map data items that describe driving conditions into a collision classification. We used these reports to help determine circumstances under which self-driving cars carry out actions that cause collisions or fail to avoid them. These harmful actions are referred to as anti-autonomy traits and factors on the part of the vehicle that cause collisions and diminish trust [3][39]. Data availability was a challenge since jurisdictions of different states, federal traffic agencies and motor vehicle departments often do not make their data publicly available. Data used in this study was submitted by the manufacturers of autonomous vehicles to the California Department of Motor Vehicles for collisions that occurred with other cars, pedestrians, bicyclists, and other objects during their test drives on roads and freeways in the state. All data applied to collisions that occurred while the cars were being driven in autonomous driving mode. The collisions occurred between October 2014 and March 2020 [3][39]. The attributes of this dataset are listed in Table I below. All attribute names are feature type categorical and data type object.

TABLE I. COLLISION DATA ATTRIBUTES [3]

Attribute Type	Attribute Names
Autonomous vehicle details	Manufacturer Name, Business Name, Vehicle Year, Vehicle Make, Vehicle Model, Vehicle was (stopped in traffic/moving)
Accident Details	Date of Accident, Time of Accident
Involved in Autonomous vehicle accident	Involved in Autonomous Vehicle Accident (Pedestrian/Bicyclist/Other), Number of vehicles involved with Autonomous Vehicle
Autonomous vehicle damage	Vehicle Damage, Damaged Area
Details of other vehicle involved in accident	Vehicle 2 Year, Vehicle 2 Make, Vehicle 2 Model, Vehicle 2 was (stopped in traffic/moving)
Involved in other vehicle accident	Involved in Vehicle 2 Accident Pedestrian, Involved in Vehicle 2 Accident Bicyclist, Involved in Vehicle 2 Accident Other, Number of vehicles involved with Vehicle 2
Injuries	Injured, Injured Driver, Injured Passenger, Injured Bicyclist
Vehicle driving mode	Vehicle Driving Mode
Weather conditions for both vehicles	Clear, Cloudy, Raining, Snowing, Fog/Visibility, Other, Wind
Lighting conditions for both vehicles	Daylight, Dusk-Dawn, Dark Street Lights, Dark-No Street Lights, Dark-Street Lights Not Functioning
Roadway surface for both vehicles	Dry, Wet, Snowy-Icy, Slippery/Muddy/Oily/etc., Holes-Deep-Rut, Loose Material on Roadway, Obstruction on Roadway, Construction/Repair Zone, Reduced Roadway Width, Flooded, Other, No Unusual Conditions
Preceding Movement of Autonomous Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Preceding Movement of Other Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing Into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Type of Collision	Head On, Side Swipe, Rear End, Broadside, Hit Object, Overturned, Vehicle/Pedestrian, Other
Other	CVC Sections Violated Cited, Vision Obscurement, Inattention, Stop and Go Traffic, Entering/Leaving Ramp, Previous Collision, Unfamiliar with Road, Defective WEH Equip Cited, Uninvolved Vehicle, Other, Non-Apparent, Runaway Vehicle

This data was extracted from PDF files and converted into CSV format with 256 rows of data in 140 columns. As is often the case in machine learning, data cleaning was a significant effort, and included pre-processing steps for augmenting, labeling, and classifying the data [3][39]. Data was augmented to 5256 rows with the goal of yielding a model with minimal noise. Details of data augmentation criteria is described in [3]. The core purpose of the study was to associate conditions into a level of trust that people had in a self-driving car. The values of attributes in the data that describe conditions and circumstances that are present when a collision occurs provide a handle to model a mapping between data and trust level. After pre-processing the data, a linear sequential supervised learning ANN model called NoTrust was devised, validated, and tested to classify the target data, using the basic approach illustrated in Figure 1.

The model used the libraries provided by Keras with the Tensorflow backend [40][41][42]. Python was used for programming since it integrates with Keras to access the neural network Application Programming Interface. The deep learning libraries of Keras support fast prototyping, modularity, and smooth computation.

There are multiple challenges concerning data, features, and metrics in applying the ML methodology to the self-driving car application. First, there were only 256 collision reports available, which is arguably a small number to use in a ML method. This was mitigated by augmenting the data to produce a larger set to develop a model with higher accuracy. Also, in the context of alternative target value possibilities, the data is unbalanced in that the number of samples across the distinct classes differs widely. Section V describes methods, such as oversampling, to deal with unbalanced data. Second, there are 140 attributes, a large number relative to sample size, as shown in Table I. Thus, the possible permutations and combinations that could be evaluated in the ANN model are explosive. Fortunately, with initial analyses of the data and evaluation runs, it was possible to identify a subset of attributes and features that were mandatory to include. This analysis was done by systematically configuring sets with and without specific attributes, evaluating each combination, and comparing the outcomes. Using this search method, we identified certain categorial features that were closely correlated with the anti-autonomous traits of the vehicles. This supported dropping the features related to date/time of accident, vehicle manufacturer, weather conditions, lighting conditions, roadway conditions, vehicle movement and type of collision. Ultimately, we arrived at the five attributes shown below to form the mandatory set.

- Vehicle driving mode = autonomous
- Vehicle damage = moderate and major
- First vehicle involved = Pedestrians/Bicycle/Other
- Second vehicle involved = Bicycle/Other
- Injuries sustained = Pedestrians/Bicyclists/Others

While keeping the model simple and still retaining accuracy, the mandatory feature set performed well in making trust and do not trust predictions for autonomous vehicles. However, when anti-autonomous traits of the self-driving car itself were incorporated into the model it became apparent that more attributes had to be utilized.

Anti-autonomy refers to decisions and actions taken by a self-driving car that are in some way inappropriate in terms of increasing risk, diminishing safety, causing potential harm, or lowering trust. It entails from an unexpected, unconventional, and abnormal decisions that self-driving car makes in the event of unfavorable surrounding driving conditions related to weather, roadway surface conditions, drivability of other vehicles and, pedestrians or bicyclists sharing the road. Autonomous vehicles have been known to have certain technological shortcomings in terms of Lidar failing when the weather conditions are rainy or foggy with limited visibility. Extensive study and analysis of the collision data revealed the following anti-autonomous behavior of self-driving cars [3]:

- In 50.22% of collisions, a pedestrian was involved while the vehicle was driving autonomously.
- In 52.08% of collisions, a pedestrian was involved while vehicle was driving autonomously and was moving in traffic.
- In 55.13% of the collisions, a bicyclist was involved while the vehicle was driving autonomously and was about to slow down or stop. These statistics illustrate the confusing behavior of a self-driving car, its mechanical functioning and decision making at the second when a bicyclist appears in front of them.
- In 55.84% of the collisions, a pedestrian was involved while the vehicle was driving autonomously and was attempting to make a parking maneuver. These statistics reveal a potential malfunction of vehicle operation in terms of gauging pedestrian behavior and hitting the breaks just in time.
- 54.28% of the collisions happened when the vehicle was driving autonomously during foggy weather with limited visibility. These statistics depict the malfunctioning of the sensors, camera, and Lidar sensors.
- In 51.71% of the collisions, a pedestrian was involved while the vehicle was driving autonomously during foggy weather with limited visibility.
- In 50.68% of the collisions, a bicyclist was involved when the vehicle was driving autonomously at night when streetlights were on.
- In 53.18% of the collisions, a pedestrian was involved when the vehicle was driving autonomously at night when streetlights were on.

The anti-autonomous traits that were incorporated into the model include vehicle driving mode, type of collision, weather conditions, roadway surface conditions and injuries sustained by pedestrian/bicyclists/others. In addition to the linear sequential ANN, evaluation of Recurrent Neural Networks (RNN) models with Long Short-Term Memory (LSTM) were available for possible comparison purposes. The primary reason for choosing a sequential ANN model was that the classification sought is binary, predicting whether the autonomous vehicles could be trusted. A sequential ANN model utilizes a stack of layers with each layer having exactly one input tensor and one output tensor. This contrasts with a Functional API with shared layers, non-linear topology, and multiple inputs/outputs. This study uses an input stack of layers of selective features that have a single output to model affirmation or denial of trust. Thus, we avoided layer sharing, non-linear topologies, multiple inputs/outputs, or creation of a directed acyclic graph or graph of layers. These properties favor the choice of sequential ANN. Also, this model was chosen over RNN because the data utilized for model processing was not a time series or natural language sequence data. When the additional attributes are included, along with measures of severity of damage sustained by vehicle, the imbalance of the data increases. More specifically, the larger number of predictors added more noise, redundancies, increased overfitting, and decreased the quality of the predictions. A related study by Meiri and Zahavi [4] used simulated annealing to search the attribute space.

Combinatorial problems often have issues related to model accuracy, performance, and optimizer bias. Also, the model solutions offered by machine learning include approximation errors, which further exacerbates issues related to differences between training and validation data [5]. This can be solved by two approaches – active learning and passive learning. Active learning involves updating the model itself to assure a convergence between training and validation curves, in turn improving model accuracy and optimization bias. Passive learning involves the training set providing a uniform and sufficient coverage of the search space [5]. In a similar context, Charikar, Guruswami, Kumar, Rajagopalan, and Sahai [6] defined and studied combinatorial feature selection problems, presented a theoretical framework, and provided algorithms on approximation and hardness results of these combinatorial problems [6].

IV. INTRUSION DETECTION APPLICATIONS

In today's world of connected devices, security of the network is of critical importance. Unauthorized access and malicious activities are a great threat to confidentiality, integrity, and availability that form the information security triad. The role of an Intrusion Detection System (IDS) is to detect abnormalities caused by an unauthorized reach into the network and send alerts. An IDS is an element of support for a wall of defense between cyber-attacks. Supervised ML techniques in an IDS can provide high detection accuracy, particularly against known types of attacks.

The NSL-KDD is an update and improvement to the KDD'99 dataset that was developed for the KDD Cup competition in 1999 [7]. These datasets are publicly available and are very widely used for IDS experiments. The data is primarily internet traffic consisting of 43 features per record, of which the last two are *class* (attack or normal) and *score* (severity of traffic input) [8]. The *class* column provides information on whether the record is considered normal or is a member of one of four attack classes - Denial of Service (DoS), Probe, Remote-to-Local (R2L) or User-to-Root (U2R). There are14 attack types under these 4 classes: Apache2, Smurf, Neptune, Back, Teardrop, Pod, Land, Mailbomb, Processtable, UDPstorm, WarezClient, Guess_Password, WarezMaster, Imap, Ftp_write, Named, Multihop, Phf, Spy, Sendmail, SmpGetAttack, AnmpGuess, Worm, Xsnoop, Xlock, Buffer_Overflow, Httptuned, Rootkit, LoadModule, Perl, Xterm, Ps, SQLattack, Satan, Saint, Ipsweep, Portsweep, Nmap, Mscan [43][44]. A mixture of categorical (nominal), binary and numeric variables are in the feature set. Each record has basic, content-related, time-related, and host-based features [9]. The attributes of this dataset are listed in Table II.

TABLE II. NSL-KDD DATASET ATTRIBUTES [9]

Attribute Type	Attribute Names
Basic	Duration, Protocol_type, Service, Flag, Src_bytes, Dst_bytes, Land, Wrong_fragment, Urgent
Content related	Hot, Num_failed_logins, Logged_in, Num_compromised, Root_shell, Su_attempted, Num_root, Num_file_creations, Num_shells, Num_access_files, Num_outbound_cmds, Is_hot_login, Is_guest_login
Time related	Count, Srv_count, Serror_rate, Srv_serror_rate, Rerror_rate, Srv_rerror_rate, Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate
Host based traffic	Dst_host_count, Dst_host_srv_count, Dst_host_same_srv_rate, Dst_host_diff_srv_rate, Dst_host_same_src_port_rate, Dst_host_srv_diff_host_rate, Dst_host_serror_rate, Dst_host_srv_serror_rate, Dst_host_rerror_rate, Dst_host_srv_rerror_rate

The study also used the UNSW-NB15 dataset. This dataset has 49 features categorized into 6 groups: basic, flow, time, content, labelled and additional generated features [10]. There are 9 attack types: fuzzers, analysis, back-doors, DoS, exploits, generic, reconnaissance, shell code and worms [11]. This dataset has a mixture of categorical, binary, and numerical datatypes. The attributes of this dataset are listed in Table III below.

TABLE III. UNSW-NB15 DATASET ATTRIBUTES [16]

Attribute Type	Attribute Names
Basic	state, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, service, sload, dload, spkts, dpkts
Flow	srcip, sport, dstip, dsport, proto

Content	swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len
Time	sjit, djit, stime, ltime, sintpkt, dintpkt, tcprtt, synack, ackdat
Additional generated (general purpose)	is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd
Additional generated (connection)	ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm
Labelled	attack_cat, attack_cat

The target attribute either identifies records as 'normal' or 'attack' or distinguishes the record as a particular attack type. Depending on the desired goal of an intrusion detection system, the machine learning model is tuned to identify a particular attack, which is a challenge. It is thus essential to understand the requirement thoroughly and preprocess input data accordingly.

As an illustration of evaluation metrics, at a high level in the IDS study, for each input vector we have exactly one of the following outcomes:

TP = True Positive = Correct predication that the input vector is an Attack

TN = True Negative = Correct prediction that the input vector is not an Attack

FN = False Negative = Incorrect prediction the input vector is not an Attack

FP = False Positive = Incorrect prediction the input vector is an Attack

The most widely reported metric is Basic Accuracy of the model, which simply reports the proportion of attack reports that are correct.

$$Accuracy = (TP + TN)/(TP = TN = FP + FN)$$

Basic Accuracy is notoriously deceptive when the classes are unbalanced, as in the case of intrusion detection studies, where most input vectors are not attacks. False reports are of interest. This gives rise to the need for metrics such as Precision and Recall, which can be calculated from information in the confusion matrix given below.

	Prediction		
Actual		Attack	Not an Attack
	Attack	TP	FN
	Not an Attack	FP	TN

$$Precision = TP/(TP + FP)$$

$$Recall = TP/(TP + FN)$$

Precision measures the proportion of the vectors reported by the IDS as attacks that are real attacks. Recall measures the proportion of the vectors that are real attacks and do get reported as such. This means that when Recall is high the IDS does not misclassify many true attacks.

In Intrusion Detection applications, false negatives can be very deadly, which favors high Recall. However, dealing with false positives also has a cost. Unfortunately, experiments that improve Precision typically reduce Recall. The reverse is also often true. For this reason, the harmonic mean of the two, called the F1 score is often calculated.

$$F1 = (2*(Precision * Recall))/(Precision + Recall).$$

The effect of the F1 score, which falls between 0 and 1, is to punish extreme values.

The NSL-KDD and UNSW NB-15 datasets are used for training machine learning models by several researchers. In [34] researchers trained the KDD-99 dataset with a mutation of a convolutional neural network and Long Short Term Memory (LSTM) network. The machine learning algorithm test accuracy was 99.7% , which outperformed other models, including DenseNet, CNN (Convolutional Neural Network), GRU (Gated Recurrent Unit), BiLSTM, and Auto Encoder. The experiment was multilabel and nearly all the individual target variables had f1-score, precision and recall exceeding 98%. Work reported in [54] and [55] took similar steps for setting up a machine learning model experimental design to train the NSL KDD and UNSW NB-15 datasets and were able to improve accuracy in both cases.

The NSL-KDD data set has 49 attributes of 6 types and 9 types of intrusions. We considered possibilities for reducing the number of attributes without eliminating information critical to the classifications. Principal Component Analysis (PCA) is a time-honored statistical method for identifying high correlations and ranking the relative importance of the attributes. Although we considered PCA, we alternatively chose to implement an autoencoder neural network. In brief, an autoencoder is a multiple layer neural network in which the input and output layers are identical, and a middle layer is of smaller dimensionality. An autoencoder is a deep learning unsupervised learning method in that the labels for the known data play no role, and after training, the middle layer becomes a compressed version of the input data. The middle layer serves as a pattern that is a discrete model of the data in a compact form. The approach requires experimentation that searches through alternative topologies and tuning parameters of the neural network, including number of layers, nodes within layers, learning rate, and number of epochs. Results of the autoencoder experimental work are given in Table IV below.

TABLE IV. PERFORMANCE EVALUATION OF IDS CLASSIFICATION

Algorithm	Accuracy (%)	Precision	Recall	F1_score
Autoencoder	88.76	0.852	0.971	0.908
SVM	86.54	0.824	0.913	0.901
Logistic Regression	82.12	0.808	0.921	0.893

The results show that the model performs well as a binary classifier for threat detection. After applying auto encoder, the accuracy reaches 88.76%, False positives occur at a low rate, so that the method flags only a few normal network inputs as attacks, , which is useful in keeping a focus on real threats. Importantly, recall at a high level of 97.1% means that nearly all real attacks are correctly identified and flagged. This is a must-have feature of an IDS since undetected attacks can be very damaging. SVM with RBF kernel (Radial Basis Function) competes with the Autoencoder with a good F1 score. The simple logistic regression works well for binary classification but fails to achieve better results for the NSL KDD dataset. These experiments are a baseline for machine learning methodologies and their proper application. We did not perform any parameter tuning for this experiment, which might have improved the accuracy and performance metrics.

V. DATA AND FEATURE ENGINEERING

We provide descriptions of data management and feature engineering issues that are pervasive in ML practice and were of importance in our applied studies.

A. Data Management

Class imbalance in a dataset means that the relative numbers of instances within the classes vary significantly in number [17]. The magnitude of the discrepancies will also vary. Class imbalance is common in most important data domains, including detection of things like fraudulent activities, anomalies, oil spills, and in medical diagnoses. The imbalance of classes occurs in both binary class and multi-class classification [18]. In binary classes, the smaller and larger cardinality classes are called minority and majority classes respectively [17][19]. Class imbalance can influence the training process of ML techniques and lead to ignoring the minority class entirely. We discuss some of the approaches to treat class imbalances. Figures 4 – 10 illustrate the results of applying each technique.

Random oversampling of a minority class. In this approach data instances in minority classes are duplicated at random to induce a balance of membership between classes. Due to randomness of the oversampling, the method is naïve in that it makes no assumptions about the classes and their members [20][21]. Since exact copies of some data instances are included in training, there is a risk of overfitting. Classifier accuracy may also be influenced, and computational effort may be increased.

Random undersampling of a majority class. This approach discards data instances from majority classes to induce balancing [22]. As in the case of the oversampling method, the discarded data is chosen randomly and naively. The method applies to both binary and multiclass data. The approach can make it difficult to distinguish boundaries between classes, with an inimical impact on performance measures [23].

Synthetic Minority Oversampling Technique (SMOTE). This technique was introduced in 2002 to address the shortcomings of the oversampling and undersampling approaches [24][25]. The technique generates synthetic data by calculating feature space similarities between minority class data instances. The K-nearest neighbors of each data instance in a minority class are calculated, then randomly selected one by one. The method then calculates linear interpolations among the data and uses them to create synthetic data instances.

Borderline SMOTE. The SMOTE approach encounters issues when minority class data instances occur in the vicinity of majority class data instances, creating undesirable bridges. The Borderline SMOTE variation addresses this drawback by oversampling only minority class instances near the borderline. Data points are called border points if they are incident to both minority and majority classes and called noise points otherwise [26]. Border points are then utilized to balance the data between classes.

K-Means SMOTE. This technique generates minority class samples in safe and crucial borders of input spaces and thus assists performance in classification. The method begins by clustering the dataset using the K-means procedure, then selects the clusters that have higher numbers of minority samples [27]. Additional synthetic samples are then assigned to clusters where minority class samples are sparsely distributed. No noise points are generated.

SVM SMOTE. A variation of Borderline-SMOTE, the method finds misclassification points. The borderline points are approximated and classified with a Support Vector Machine (SVM) classifier [28]. Synthetic data points are created randomly around the lines joining each minority class support vector with its neighbors.

Adaptive Synthetic Sampling – ADASYN. A limitation of Borderline SMOTE is that is utilizes only synthetic points generated from extreme observations and the borderline instances and neglects the rest of the points in minority classes. This issue is addressed by ADASYN by creating synthetic data using the density of the existing data [29]. The ratio of synthetically generated data is created in inverse relation to the minority class density. In this way, a less dense area creates more synthetic data.

The Churn Modeling Data from Kaggle was applied to the methods [30]. Figure 4 shows the distribution of the data in the original classes, followed by the outcomes of the alternative methods in Figures 5 to 10.

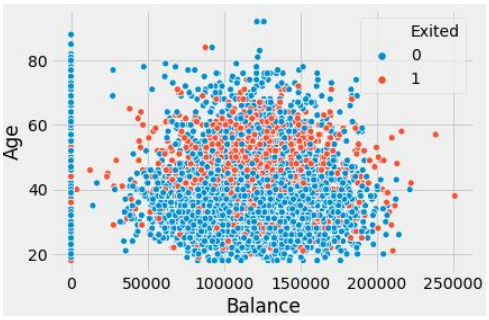


Figure 4. Original Class Imbalance Illustration.

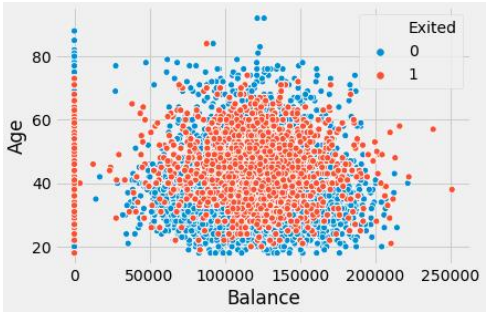


Figure 5. Outcome of Random Oversampling.

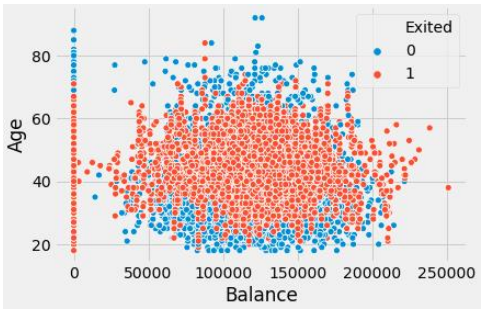


Figure 6. Outcome of SMOTE.

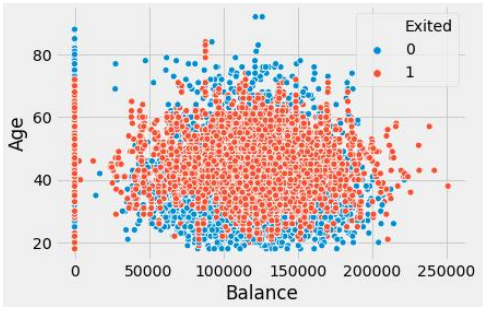


Figure 7. Outcome of Borderline-SMOTE.

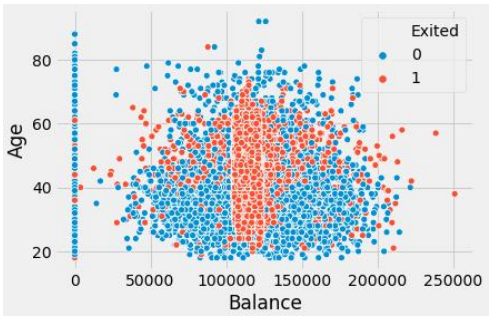


Figure 8. Outcome of K-Means SMOTE.

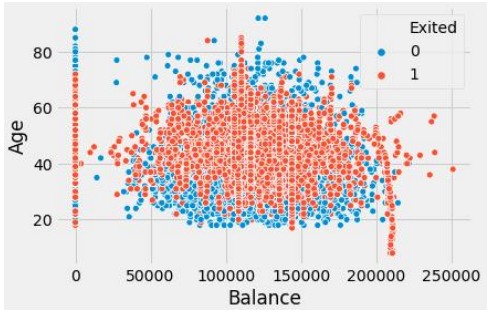


Figure 9. Outcome of SVM SMOTE.

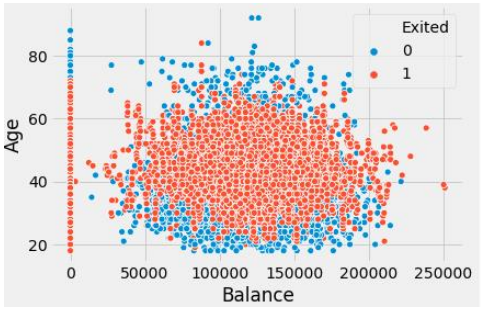


Figure 10. Outcome of ADASYN.

B. Data Management and Feature Engineering

There are multiple methods for feature engineering on categorical data. The inputs to ML algorithms must be numeric, but many applications have categorical data. In our work with ML methods for self-driving car collisions there are examples of ordinal categorical data, such as rating of severity of damages or a weather condition. The intrusion detection work examples include counts of file access attempts, session duration, or error rates. There are also examples of nominal data, such as a type of vehicle in our self-driving car work, or whether a flag is set in the intrusion detection work. Various encoding methods are used to convert the variables into a useful numerical representation [10]. Choosing an appropriate encoding scheme is an essential part of data preprocessing for a ML algorithm. Some of the categorical encoding techniques are described below.

B.1. One-hot encoding

This method converts an attribute with N possible categories into N distinct features. In the NSL-KDD dataset, the *protocol type* attribute has 3 possible values - Internet Message Control Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). One-hot encoding converts this attribute into three feature columns as shown in Figure 11. It follows a 0/1 representation to indicate presence (indicated by 1) or absence (indicated by 0) of a value.

protocol_type
icmp
udp
tcp
tcp
udp

⇒

icmp	udp	tcp
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0

Figure 11. One-hot encoding used in Protocol Type attribute.

One-hot encoding has the advantage that it can convert ordinal and categorical features into orthogonal vector space [56]. It is one of the most used feature transformation techniques in machine learning experiments. The disadvantage of one-hot encoding appears when features with high cardinality are transformed. It might have the curse of dimensionality, also leading to a very sparse matrix. Also, there is a possibility to introduce bias in case the feature is distributed in an orderly fashion [57].

B.2. Dummy coding

Like one-hot encoding, dummy coding converts an attribute with N values to a feature set of N-1 values. The converted set of binary variables are called dummy variables. Figure 12 illustrates one-hot ending and dummy coding applied to the same set of categorical records.

Column	Code
A	10
B	01
C	00

One-Hot Coding

Column	Code
A	10
B	01
C	00

Dummy Code

Figure 12. One-hot and dummy encoding used in the same dataset [12].

An advantage of dummy encoding is that it decreases the complexity and entropy complications. Dummy variable is of slightly better space complexity since it creates (k-1) dummy variables for k original variables. Whereas one hot encoding creates exactly k number of converted variables. The dummy variable trap is a camp term in machine learning preprocessing that refers to more than one independent categorical feature being multi-collinear [58][59].

B.3. Effect coding

While one-hot encoding and dummy coding use only 0 and 1 to encode categorical variables, effect coding sets values that sum to zero in the new feature set. As a result, negative values may also be generated in the encoded feature set. Effect coding is a preferred choice when there is an interaction of categorical variables in a dataset as it can provide reasonable estimates of main effect and of the interaction [13]. Effect coding has an advantage over dummy coding when there is an interaction between categorical variables [60]. The benefit is that the feature will achieve a reasonable estimate of both main effect and interaction using effect coding, which is efficient when training data is unbalanced [61].

B.4. Hash encoding

Hash encoding is appropriate for categorical variables that have many possible values. The method uses a hash function to map categorical values into numbers. Commonly used hash functions include Message Digest functions MD2, MD4, MD5 and Secure Hash Algorithms SHA0, SHA1, SHA2 and SHA3. The MD5 hash function is used by default [12]. Hash encoding returns a variable map with smaller dimensions than other encoding schemes, such as one-hot encoding or dummy coding. Figure 13 below illustrates the hash-encoding process:

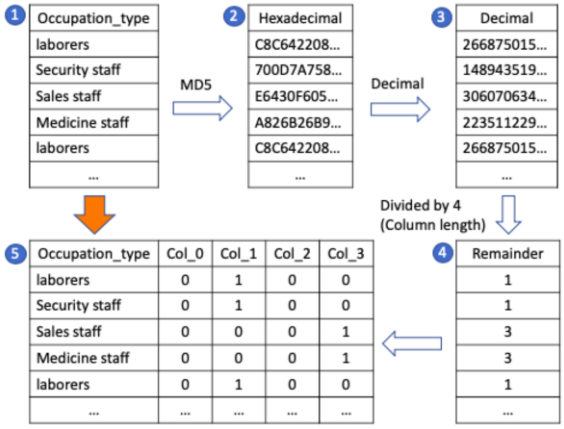


Figure 13. Hash Encoding Process [14].

Hash encoding is a fast and memory-efficient technique for categorical feature encoding. It has limitations such as inducing collisions and losing ordering information. Unlike the one-hot and dummy encoding, the method can handle unseen categorical values during interference by mapping them to a random numerical value [62]. Also, since it does not need a separate encoding dictionary, it is quite easy to deploy. Hash encoding can result in collisions, where different categorical values are mapped to the same numerical value. This can reduce the accuracy of the machine learning model and make it difficult to interpret the

importance of the feature. Hash encoding is considered unsuitable for categorical features with a small number of unique values, since the collision rate can be high, and the encoding may not be informative [63].

B.5. BaseN encoding

BaseN encoding converts categorical variables into a consistently encoded feature set using a selected base, such as base 2 for binary encoding. The base or radix is the available number that can be used in different combinations to represent all values in a numbering system. A BaseN encoder encodes categorical values into arrays of their base-n representation.

BaseN encoding is a powerful technique for encoding categorical variables as numerical variables. BaseN encoding preserves ordinal relationships among categories, which can improve the interpretability and accuracy of the machine learning model [64]. It can reduce the number of features in the machine learning model, which improves computational efficiency and reduces risk of overfitting. Limitations are potential for noise or sparsity, inability to capture complex relationships, and compatibility with specific machine learning models. The suitability of BaseN encoding depends on the specific dataset and machine learning task at hand [65].

B.6. Target encoding

Target encoding (also known as mean encoding) replaces a variable with a mean of the target value for that variable. Figure 14 provides an illustration. When the values of a categorical variable are already of a high volume, target encoding provides an advantage over other methods as it does not add extra dimensions to the dataset.

	feature	feature_label	feature_mean	target
0	Moscow	1	0.4	0
1	Moscow	1	0.4	1
2	Moscow	1	0.4	1
3	Moscow	1	0.4	0
4	Moscow	1	0.4	0
5	Tver	2	0.8	1
6	Tver	2	0.8	1
7	Tver	2	0.8	1
8	Tver	2	0.8	0

Figure 14. Target Encoding [15].

Target encoding can reduce the number of features in the machine learning model, which improves computational efficiency and reduces the risk of overfitting [66]. Target encoding is a powerful technique for encoding categorical variables as numerical variables based on the target variable. Limitations include potential for bias or overfitting, limited generalization to new data, and lack of compatibility with certain machine learning models. The suitability of Target encoding depends on the specific dataset and machine learning task. It is recommended to use cross-validation to avoid overfitting when using target encoding [67].

B.7. Label or ordinal encoding scheme

Ordinal categorical variables require that the order of the variable be preserved. For example, a road surface when a collision occurs might be categorized as dry, somewhat wet, or very wet so that the 3 values have an order that might provide additional insights. Ordinal encoding scheme aims at preserving this order when mapping values to numeric form. The method simply assigns each label a number (for example dry=1, somewhat wet=2 and very wet=3).

Label encoding preserves ordinal relationships among categories, which improves the interpretability and accuracy of the model [68]. It can reduce the number of features in the machine learning model, improving computational efficiency, and reducing risk of overfitting. Limitations are potential for bias or arbitrary ranking, inability to capture complex relationships, and lack of compatibility with specific machine learning models. The suitability of Label encoding depends on the specific dataset and machine learning task addressed. It is recommended to use label encoding only when the categorical variable has a meaningful order, or the number of categories is small [69].

C. Feature Selection

The complexity of ML models increases with the dimensionality of the dataset. Predictive models often fail to achieve high accuracy because of inadequate analyses directed to feature selection. Selecting the most important and significant features reduces the complexity of the model and can also increase the prediction performance [37][38]. Multiple approaches are available and effective for reducing the feature set. Prominent ones are described below.

Filter Methods. In our work, we choose feature selection methods that apply to situations with a categorical output, such as whether an input vector is an attack or not. The filter methods eliminate features independently of the ML method used. A univariate feature filter evaluates the importance of single features using univariate statistical tests. Each feature is paired with the target to evaluate statistical significance between them. The analysis of variance or ANOVA F-test is widely used. The F-test calculates the ratio between variance values [31]. The resultant measures of the relative importance of individual features provide a tool for determining features that are unnecessary or of little importance.

The filter method is effective for scalability, especially for high-dimensional datasets with a large number of features [70]. It improves accuracy and interpretability by removing irrelevant and redundant features. It also helps to reduce overfitting and improve generalization. The disadvantages of the filter method are the lack of modeling interaction and/or non-linear relationships between features and the target

variables [79]. The method limits the accuracy and predictive power to a certain threshold [71].

Wrapper Methods. Wrapper methods directly evaluate combinations of features by running the ML model restricted to the set of candidate features. Taken to an extreme, all combinations would be evaluated, an impossible task in practice. Thus, various search space approaches are employed. Forward search iteratively adds promising feature vectors one by one to build a feature set. Backward search starts with all features and successively eliminates those that perform poorly. Many approaches based on optimization techniques are available [32][33]. The self-driving car work basically follows a forward selection approach based on both advance knowledge about the importance of certain features from analytics on the data itself along with test runs of the ML model. Heavy computational load and possibilities of overfitting are potential drawbacks [34].

The Wrapper method is efficient in capturing complex relationships and interactions between features and targets. It can handle non-linear relationships which is often impossible for other feature selection techniques [72]. Like the filter method, it improves the interpretability and generalization ability of the model. Wrapper methods are computationally expensive, especially for high dimensional data. Also, the wrapper method can overfit the model to the training data, especially if the search space of feature subsets is large or the evaluation metric is not carefully chosen [73].

Embedded Methods. Embedded methods utilize mathematical information that is available during the training of the model to determine the relative importance of features. In some sense embedded methods mitigate the drawbacks of the filter and wrapper methods but retain their strengths. When implemented carefully, they are not prone to overfitting [35]. The XGBoost technique produces an importance score for each attribute that is used to identify those that can be confidently eliminated [36]. In applications like intrusion detection, having many attributes presents a huge computational burden. The embedded methods are highly successful in greatly reducing the features needed in intrusion detection ML work.

The Embedded method handles non-linear relationships and interactions among features and the target variable, a capability that may not possessed by the filter method or many other feature selection techniques [74]. The method is a powerful technique for feature selection that can capture complex relationships and interactions among features and the target variable. Limitations include high computational cost, lack of compatibility with certain models, and potential for overfitting or underfitting. The suitability of embedded method depends on the specific dataset, machine learning model, and optimization algorithm [75].

VI. CONCLUSION AND FUTURE WORK

Machine learning is now a well-established and effective approach in many domains. When using machine learning approaches in practice, issues arise concerning choices for which type of model to use, parameters to choose and tune, data management, feature engineering and selection. We address many of these issues in the context of applications to self-driving cars and intrusion detection. The applications are of high importance in inter-related areas of cybersecurity, trust, risk, safety, reliability, autonomy, and anti-autonomy. For the studies concerning self-driving cars, we present uses for reinforcement learning and supervised learning that reveal circumstances under which the autonomous vehicle makes decisions to take actions that result in collisions. For intrusion detection, model choices and computational results are presented that result in excellent values for performance metrics. Included are reports of recall metrics that indicate that it is possible to use machine learning methods that flag nearly all potentially dangerous attack vectors. The data-centric and feature engineering challenges are extensive and detailed, but addressable. We describe approaches to addressing these challenges. Results reveal several implications for needs for next steps in research. New frontiers include methods that can be deployed in real-time, automate feature engineering, choose, and extract features dynamically, and simultaneously support optimization of multiple performance evaluation metrics.

REFERENCES

[1] K. E. Nygard, M. Ahsan, A. Rastogi, and R. Satyal, “Data and Feature Engineering Challenges in Machine Learning,” in *ADVCOMP 2022, The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 1–10, November 2022.

[2] T. Hey, S. Tansley, and K. Tolle, “The Fourth Paradigm: Data-Intensive Scientific Discovery,” Redmond, Washington: Microsoft Research, 2009.

[3] A. Rastogi, “Trust and Anti-Autonomy Modelling of Autonomous Systems,” Order No. 28255688, North Dakota State University, Ann Arbor, 2020.

[4] R. Meiri and J. Zahavi, “Using simulated annealing to optimize the feature selection problem in marketing applications,” In *European Journal of Operational Research*, vol. 171, no. 3, pp. 842-858, 2006.

[5] M. Lombardi and M. Milano, “Boosting combinatorial problem modeling with machine learning,” In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5472-5478, 2018.

[6] M. Charikar, V. Guruswami, R. Kumar, S. Rajagopalan, and A. Sahai, “Combinatorial feature selection problems,” In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 631-640, 2000.

[7] KDD Cup 1999 Data, Kdd.ics.uci.edu [Online]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed: 2022.08.22].

[8] A Deeper Dive into the NSL-KDD Data Set. Medium [Online]. Available from: https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657 [Accessed: 2022.08.22].

[9] L. Dhanabal and S.P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," In *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446-452, June 2015.

[10] S. Choudharya and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," In *Procedia Computer Science*, vol. 167, pp. 1561-1573, 2020.

[11] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), IEEE*, pp. 1-8, 2018.

[12] 8 Categorical Data Encoding Techniques to Boost your Model in Python!, Analytics Vidhya [Online]. Available from: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/ [Accessed: 2022.08.22].

[13] Introduction to SAS, UCLA: Statistical Consulting Group. [Online]. Available from: https://stats.idre.ucla.edu/sas/modules/sas-learning-moduleintroduction-to-the-features-of-sas/ [Accessed: 2022.08.22]

[14] A Data Scientist's Toolkit to Encode Categorical Variables to Numeric. [Online]. Available from: https://towardsdatascience.com/a-data-scientists-toolkit-to-encode-categorical-variables-to-numeric-d17ad9fae03f [Accessed: 2022.08.22]

[15] Improve your classification models using Mean /Target Encoding. [Online]. Available from: https://medium.com/datadriveninvestor/improve-your-classification-models-using-mean-target-encoding-a3d573df31e8 [Accessed: 2022.08.22]

[16] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," In *2015 military communications and information systems conference (MilCIS), IEEE*, pp. 1-6, 2015.

[17] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," In *Intelligent data analysis*, vol, 6, no. 5, pp. 429-449, 2002.

[18] R. Gomes, M. Ahsan, and A. Denton, "Random Forest Classifier in SDN Framework for User-Based Indoor Localization," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0537-0542, 2018.

[19] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," In *Int. J. Adv. Soft Comput. its Appl.*, vol. 5, no. 3, 2013.

[20] A. Ghazikhani, H. S. Yazdi, and R. Monsefi, "Class imbalance handling using wrapper-based random oversampling," In *20th Iranian Conference on Electrical Engineering (ICEE2012), IEEE*, pp. 611-616, 2012.

[21] Z. Zheng, Y. Cai, and Y. Li, "Oversampling method for imbalanced classification," In *Computing and Informatics*, vol. 34, no. 5, pp. 1017-1037, 2015.

[22] A. M. Denton, M. Ahsan, D. Franzen, and J. Nowatzki, "Multi-scalar analysis of geospatial agricultural data for sustainability," In *2016 IEEE International Conference on Big Data (Big Data), IEEE*, pp. 2139-2146, 2016.

[23] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special Issue on Learning from Imbalanced Data Sets," In *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1-6, 2004.

[24] M. Ahsan, R. Gomes, and A. Denton, "SMOTE Implementation on Phishing Data to Enhance Cybersecurity," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0531-0536, 2018.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," In *Journal of artificial intelligence research*, vol. 16, pp. 321-357, 2002.

[26] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," In *International conference on intelligent computing*, pp. 878-887. Springer, Berlin, Heidelberg, 2005.

[27] H. Hairani, K. E. Saputro, and S. Fadli, "K-means-SMOTE untuk menangani ketidakseimbangan kelas dalam klasifikasi penyakit diabetes dengan C4.5, SVM, dan naive Bayes," In *J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

English – H. Hairani, K. E. Saputro, and S. Fadli, "K-means-SMOTE to trate class imbalance in the classification of diabetes with C4.5, SVM, and Naïve Bayes," *In J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

[28] Y. Tang, Y. Q. Zhang, and N. V. Chawla, "SVMs modeling for highly imbalanced classification," In *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281-288, 2008.

[29] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE*, pp. 1322-1328, 2008.

[30] O. Özdemir, M. Batar, and A. H. Işık , "Churn Analysis with Machine Learning Classification Algorithms in Python," In *The International Conference on Artificial Intelligence and Applied Mathematics in Engineering*, pp. 844-852. Springer, Cham, 2019.

[31] N. O. F. Elssied, O. Ibrahim, and A. H. Osman, "A novel feature selection based on one-way ANOVA F-test for e-mail spam classification," In *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 3, pp. 625-638, 2014.

[32] M. Ahsan, R. Gomes, and A. Denton, "Application of a convolutional neural network using transfer learning for tuberculosis detection," In *2019 IEEE International Conference on Electro Information Technology (EIT), IEEE*, pp. 427-433, 2019.

[33] A. Mustaqeem, S. M. Anwar, M. Majid, and A. R. Khan, "Wrapper method for feature selection to classify cardiac arrhythmia," In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE*, pp. 3656-3659, 2017.

[34] M. Ahsan and K. Nygard, "Convolutional Neural Networks with LSTM for Intrusion Detection," In *The 35th International*

Conference on Computers and Their Applications (CATA 2018), vol. 69, pp. 69-79. 2020.

[35] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," In *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 703-715, 2019.

[36] Y. Wang and X. S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," In *arXiv preprint arXiv:1901.08433*, 2019.

[37] M. Chowdhury, J. Tang, and K. Nygard, "An artificial immune system heuristic in a smart grid," In *The 28th International Conference on Computers and Their Applications*, 2013.

[38] M. Chowdhury and K. Nygard, "Machine learning within a con resistant trust model," In *The 33rd International Conference on Computers and Their Applications (CATA 2018)*, pp. 9-14, 2018.

[39] A. Rastogi and K. Nygard, "Threat and alert analytics in autonomous vehicles," In *The 35th International Conference on Computers and Their Applications (CATA 2018)*, vol. 69, pp. 48-59, 2020.

[40] Home - Keras Documentation, Keras.io, 2020. [Online]. Available: https://keras.io/. [Accessed: 2022.08.16].

[41] "TensorFlow," TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/. [Accessed: 2022.08.16].

[42] "tensorflow/tensorflow," GitHub, 2020. [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed: 2022.08.16].

[43] M. Ahsan, N. Rifat, M. Chowdhury, and R. Gomes, "Detecting Cyber Attacks: A Reinforcement Learning Based Intrusion Detection System," In *2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 461-466, 2022.

[44] M. Ahsan, K. E. Nygard, R. Gomes, M . M . Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review," In *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 527-555, 2022.

[45] N. I. Rifat, "Feature Engineering on the Cybersecurity Dataset for Deployment on Software Defined Network," 2020.

[46] Y. Berra, "A Quote By Yogi Berra," 2022. [Online]. Available: https://www.goodreads.com/quotes/261863-it-s-tough-to-make-predictions-especially-about-the-future. [Accessed: 2022.08.22].

[47] M. Ahsan, N. Rifat, M. Chowdhury, and R. Gomes, "Intrusion Detection for IoT Network Security with Deep Neural Network," *In 2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 467-472, 2022.

[48] P. Sajda, A. Gerson, K. R. Muller, B. Blankertz, and L. Parra, "A data analysis competition to evaluate machine learning algorithms for use in brain-computer interfaces," In *IEEE Transactions on neural systems and rehabilitation engineering*, 11(2), pp. 184-185, 2003.

[49] A. Fleury-Charles, M. M. Chowdhury, and N. Rifat, "Data Breaches: Vulnerable Privacy," In *2022 IEEE International Conference on Electro Information Technology (eIT)*, Mankato, MN, USA, pp. 538-543, 2022.

[50] Z. Li, X. Ma, and H. Xin, "Feature engineering of machine-learning chemisorption models for catalyst design," *Catalysis today*, vol. 280, pp. 232-238, 2017.

[51] D. K. Davis, M. M. Chowdhury, and N. Rifat, "Password Security: What Are We Doing Wrong?," In *2022 IEEE International Conference on Electro Information Technology (eIT)*, Mankato, MN, USA, 2022, pp. 562-567.

[52] R. Vanness, M. M. Chowdhury, and N. Rifat, "Malware: A Software for Cybercrime," In *2022 IEEE International Conference on Electro Information Technology (eIT)*, Mankato, MN, USA, pp. 513-518, 2022.

[53] Y. Xu, K. Hong, J. Tsujii, and E.I.C. Chang, "Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries," In *Journal of the American Medical Informatics Association*, vol. 19, no. 5, pp. 824-832, 2012.

[54] M.Ahsan, R. Gomes, M.M. Chowdhury, and K. E. Nygard, "Enhancing machine learning prediction in cybersecurity using dynamic feature selector," In *Journal of Cybersecurity and Privacy*, vol. 1, no. 1, pp. 199-218, 2021.

[55] M.K. Ahsan, "Increasing the Predictive Potential of Machine Learning Models for Enhancing Cybersecurity," Order No. 28416096, North Dakota State University, Ann Arbor, 2021.

[56] M. K. Dahouda and I. Joe, "A deep-learned embedding technique for categorical features encoding," In *IEEE Access*, vol. 9, pp. 114381-114391, 2021.

[57] B. Gu and Y. Sung, "Enhanced reinforcement learning method combining one-hot encoding-based vectors for CNN-based alternative high-level decisions," In *Applied Sciences*, vol. 11, no.3, p. 1291, 2021.

[58] I. Mukherjee, N.K. Sahu, and S.K. Sahana, "Simulation and modeling for anomaly detection in IoT network using machine learning," In *International Journal of Wireless Information Networks*, pp. 1-17, 2022.

[59] J.M. Jerez, I. Molina, P.J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," In *Artificial intelligence in medicine*, vol. 50, no.2, pp. 105-115, 2010.

[60] R.S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," In *Advances in neural information processing systems*, vol. 8, 1995.

[61] C.H. Chien, C.C. Chang, S.H. Lin, C.W. Chen, Z.H. Chang, and Y.W. Chu, "N-GlycoGo: predicting protein N-glycosylation sites on imbalanced data sets by using heterogeneous and comprehensive strategy," In *IEEE Access*, vol. 8, pp. 165944-165950, 2020.

[62] Z. Cao, M. Long, J. Wang, and P.S. Yu, "Hashnet: Deep learning to hash by continuation," In *Proceedings of the IEEE international conference on computer vision*, pp. 5608-5617, 2017.

[63] I. Lopez-Arevalo, E. Aldana-Bobadilla, A. Molina-Villegas, H. Galeana-Zapién, V. Muñiz-Sanchez, and S. Gausin-Valle, "A memory-efficient encoding method for processing mixed-type data on machine learning," In *Entropy*, vol. 22, no.12, p. 1391, 2020.

[64] J.M Springer, C.S. Strauss, A.M. Thresher, E. Kim, and G.T. Kenyon, "Classifiers based on deep sparse coding architectures

are robust to deep learning transferable examples," In *arXiv preprint arXiv:1811.07211*, 2018.

[65] S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," In *IEEE Access*, vol. 6, pp. 48231-48246, 2018.

[66] N. Nazyrova, T.J. Chaussalet, and S.Chahed, "Machine Learning models for predicting 30-day readmission of elderly patients using custom target encoding approach," In *Computational Science–ICCS 2022: 22nd International Conference Proceedings*, vol. III, pp. 122-136, June 2022.

[67] D. Silhavy, C. Krauss, A. Chen, A.T. Nguyen, C. Müller, S. Arbanowski, S. Steglich, and L. Bassbouss, "Machine learning for per-title encoding," In *SMPTE Motion Imaging Journal*, vol. 131, no.3, pp. 42-50, 2022.

[68] A. Mottini and R. Acuna-Agost, "Relative label encoding for the prediction of airline passenger nationality," In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 671-676, 2016.

[69] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pp. 587-601, 2017.

[70] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," In *2014 science and information conference*, pp. 372-378, August 2014.

[71] A. Zheng and A. Casari, "Feature engineering for machine learning: principles and techniques for data scientists," In O'Reilly Media, Inc., 2018.

[72] S.M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," In *Computers & Security*, vol. 92, pp. 101752, 2020.

[73] M. F. Rafique, M. Ali, A. S. Qureshi, A. Khan, and A. M. Mirza, "Malware classification using deep learning based feature extraction and wrapper based feature selection technique," In *arXiv preprint arXiv:1910.10958*, 2019.

[74] J. Zhou, L. Liu, W. Wei, and J. Fan, "Network representation learning: from preprocessing, feature extraction to node embedding," In *ACM Computing Surveys (CSUR)*,vol. 55, no. 2, pp. 1-35, 2022.

[75] S. Wang, J. Arroyo, J.T. Vogelstein, and C. E. Priebe, "Joint embedding of graphs," In *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no.4, pp. 1324-1336, 2019.

[76] Introduction to Various Reinforcement Learning Algorithms. Medium [Online]. Available from: https://medium.com/towards-data-science/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287 [Accessed: 2023.03.05].

[77] A. Rastogi and K. E. Nygard, "Trust Issues in Cybersecurity and Autonomy," In *27th International Conference on Software Engineering and Data Engineering*, 2018.

[78] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and IoT applications," In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1-8, 2017.

[79] S.M. Kasongo and Y. Sun, "A deep learning method with filter based feature engineering for a wireless intrusion detection system," In *IEEE Access*, vol. 7, pp. 38597-38607, 2019.

Heatmap Weighted A* Algorithm for NPC Pathfinding and Graph Switching

Paul Williamson
School of Computing and Mathematics
University of South Wales
CF37 1DL, Pontypridd
e-mail: paul.williamson@southwales.ac.uk

Christopher Tubb
School of Computing and Mathematics
University of South Wales
CF37 1DL, Pontypridd
e-mail: christopher.tubb@southwales.ac.uk

***Abstract*— Non-Player Characters are characters within a video game, which are not controlled by a human participant. While they are mainly used to fulfil a role not designated for a human player, there are occasions when an NPC needs to play in a human role, and therefore needs to imitate appropriate gameplay behaviours, in such a way that it is not easily distinguished from a human player. Navigation is a fundamental gameplay behaviour, focused on how a player traverses the environment when undertaking objectives. This paper explores the possibility of modelling human navigation by modifying A* algorithm with a heatmap derived from human-based data. This is achieved by having participants complete a search and collect experiment. The data is saved for analysis and to develop a navigation model. NPCs using the model undertake the same experiment, but with a heatmap weighted A* graph. The experiment explores adjusting the weight of the heatmap so its influence on the pathfinding varies and a comparison can be made to see which weight better reflects the human results. This paper also investigates switching between the heatmap weighted graph and a standard A* graph, depending on the task being undertaken. This graph switching was used in an experiment to evaluate if the model has an impact on navigation perception when subjects were tasked with fighting against NPCs.**

Keywords—NPC; Player Modelling; Pathfinding; Gameplay; A* Algorithm; Perception.

I. INTRODUCTION

This paper explores using human player data to influence the way NPCs navigate the environment by manipulating a graph-based search algorithm [1]. In the context of an NPC, pathfinding is the mechanism used to find a suitable route between two points on a map. The type of game genre and size of the map can influence which technique is more practical because some solutions are only viable under predefined constraints [2]. In First Person Shooter (FPS) games, a common technique used is A* algorithm, or some variant of this method, where a 2D grid is superimposed over the map, then using cost and heuristic the algorithm calculates a shortest cost path.

This paper expands on the A* algorithm. It focuses on adjusting the weight cost of nodes in accordance with a heatmap. For the purposes of this experiment, the heatmap is generated from data captured from human players roaming the environment, undertaking an experiment in which they need to find and collect eight coins. This data enabled a model to be developed, which captures not only the general areas of navigation, but also intricate behaviours associated with the act of roaming, and the influence they have on pathfinding.

Secondly, this research uses a tagged environment to help determine pathing based decisions. This limits the distance of the routing decisions to only what is within view from the perspective of the NPC. This technique drastically reduces performance cost because, despite the size of the A* graph, the distance between the NPC and destination node is always relatively short.

In A* algorithm, graph switching refers to the process of changing from one graph representation to another during the search for a path. This can be useful in certain situations, such as when the search space becomes too large to be represented efficiently in a single graph, or when the structure of the graph changes dynamically during the search. By switching to a different graph representation, the algorithm can continue to search for a path while avoiding the limitations of the previous representation. For this research, graph switching will be used to switch between two graphs:

- Roaming: This will be the lowest priority objective; it is activated when no other tasks are being undertaken.
- Standard: This graph will be used when the NPC knows the location of its objective. This is essentially all the behaviours except roaming.

While all behaviours could be modelled and a specific graph created for each, it was decided that introducing too many graphs at a time could complicate the analysis process when determining if behavioural based graph switching is a viable technique.

In Section II, this paper discusses the motivation behind a player driven A* algorithm solution. It states why it is important for NPCs to use the same navigational behaviours as human players and why heatmaps generate a useful tool for this purpose. Section III examines research in the field of improving the usefulness of A* algorithm for pathfinding solutions, and some implications of the limitation of these methods. In Section IV, the method for the data capture experiment is explained, which involves human subjects roaming the map to collect several coins. Section V uses the same experiment as Section IV, except in this experiment, NPCs with the roaming model are used and a thorough examination of the results is conducted to determine the applicability of the model and how it compares to the human

subject results. The model evaluation in real-time is conducted in Section VI, which details how an experiment was undertaken in the form of a deathmatch between NPCs and human subjects, and then discussing the results from the perspective of how the NPCs navigation was perceived. Finally, Section VII concludes the paper.

II. Motivation

While roaming may appear a random action, it is often a more strategic behaviour where a player tries to maximise scanning efficiency by positioning their character to visually cover as much of the map as possible. This increases the likelihood they will spot their objective and reduce the chance of checking already checked areas of the map [3].

The motivation for this paper is to address how NPCs can roam the environment and increase the likelihood it will interact with a human player. This is important in both single and multiplayer games. In single player games, the game should revolve around the player, so ensuring regular engagement from NPCs is crucial. Regarding multiplayer games or roles generally reserved for a human player, it is important that NPCs can imitate the general behaviours seen in a human player, which include using roaming in a way which is consistent with the routes a player might take. For example, during a death match scenario, players roam the map in search for opponents to eliminate. When NPCs pathfinding is not modelled to reflect the same generalised routes as a human player, it can cause them to patrol areas rarely visited by players.

Heat maps offer a good overview of which parts of the map contain the most interactions. Utilising this information can help develop NPCs that are not hard coded to patrol a certain route, a technique which is commonly used, which is predictable and often recognised by a player. Instead, providing the NPC with human player acquired data so they can undertake roaming with a more human-like characteristic. This should enable naturally occurring interactions, rather than forced encounters where the NPC can appear omniscient.

The perception of omniscience is a common issue with NPCs, which is often caused by making decisions and/or performing actions with information that it should not have. For instance, in some cases an NPC will shoot at a wall with a player on the other side. They should not know the player is there. However, they are provided with an extra layer of information, which can influence actions. An important part of making NPCs appear more human-like is therefore removing this perception. The development of a new model of navigation, as discussed here, is intended to do this. To achieve this goal NPCs can only make decisions based on what they can ‘see’ and internal parameters such as health or ammunition count.

Further motivation for this research is using and modifying solutions for other pathfinding problems, specifically regarding A* algorithm. Video games handle very large A* algorithm graphics by using a variety of techniques to optimize performance. One technique is to use a grid-based system to divide the game world into smaller sections, and only perform the A* algorithm on the section of the grid that the player is currently in. This reduces the number of nodes that need to be searched and allows the algorithm to run more quickly [4]. A similar technique is to use a hierarchical version of the A* algorithm, where the game first performs a rough search of the entire game world and then zooms in to perform a more detailed search in the area of interest. This research aims to increase the practicality of A* pathfinding, by incorporating specific use graphs, which could also work with other techniques, such as hierarchical A* grid-based systems [5].

III. Background and Related Research

Pathfinding is a crucial aspect of an NPCs core mechanics. Some form of navigation is essential in games where the NPC is required to move. The complexity of the pathfinding has increased as games have become more intricate. A* algorithm has remained an important technique in modern games [3].

In FPS games, A* is popular because of its graph-based nature. It can find an optimal route between two points. However, this can lead to predictable routes, which can be exploited. Furthermore, an exponential performance cost can occur when increasing the size of the map, as it increases the size of the graph [6], thus, adding more nodes that could be checked when forming a route.

Comparison analysis was conducted by Permana et al. [7], in which they looked at A*, Dijkstra and Breadth First Search (BFS) in a maze runner genre. They focused primarily on the performance impact of each technique, as well as the efficiency in context of functionality. The results suggest that all methods are capable of pathfinding, however, A* was more efficient computationally.

There has been substantial research to modify A* so it can excel at certain tasks. Sazaki et al. [8] showed that some of the limitations of A* can be overcome by developing a model, which was used in a car racing scenario. This model focused on assisting A* with a Dynamic Pathing Algorithm (DPA). The results demonstrated that it could avoid moving obstacles. This addresses one of the problems with A*; the need to continually update the graph if the map is not static. This suggests that combining pathfinding models and techniques can yield positive results and shows that the effectiveness of A* can be enhanced when aided with other techniques.

Makarov et al. [9] used Voronoi-based pathfinding that has been developed with obstacle avoidance and tactical elements to reduce the probability NPCs will traverse the dangerous areas of the map. They showed that including what NPCs can visually ‘see’, it was able to make tactical and logistical decisions. When incorporating internal information, such as previous enemy encounters, the NPC uses all the data to make decisions, including navigation. This indicates that when making navigation decisions,

providing the NPC with more specific data about itself can lead to an adaptable NPC, which could appear more human-like.

Like the forementioned work, the research in this paper uses NPCs vision to make decisions on navigation. NPCs can only move to a location it can visually 'see'. This significantly reduces the size of the pathing and low computational overhead. When using a non-static map, the A* graph needs to be updated on a regular basis, so NPCs do not attempt to traverse non-walkable areas. This can have a negative impact on performance. While significant research has been undertaken to address this issue, it is still a problem that needs to be considered when using A*. The approach proposed in this research could be useful as the NPC could update the graph based on what is in its view.

Research undertaken by Sturtevant et al. [10] has shown that dynamically adjusting the cost of A* nodes based on the terrain they occupy can yield useful results. They used this technique by creating an abstraction layer which deals with terrain cost and dynamic terrain. They determined that from a performance perspective, when used with several different terrain types, the solution can be up to ten times faster in finding a suitable path, while remaining 2-6% optimal. This is important to this research because it shows that weighted environments can be used with other techniques to positively impact the overall pathfinding. This is supported by Pan [11] who proposes a multi-technique approach. They used a bootstrap Jump Point Space (JPS) technique when there are no threats present, then switch to a waypoint-based solution when the NPC detects a threat. This is an interesting approach to a dynamic pathfinding system which responds to the current circumstance of the NPC. When combined with a weighted A* graph, this could help develop a more realistic navigation system because the pathfinding technique will change to reflect the behaviour expected to be displayed.

Anderson [12] introduced a new heuristic method, that can be used to find the best path on a four-connected grid-world. This additive heuristic is very powerful, significantly reducing the number of nodes that need to be searched. However, it does require the use of two abstract graphs in order to calculate the heuristic. The heuristic can be calculated efficiently using on-demand techniques, with instance-dependent pattern database (IDPDB) being the most effective method. This method reduces the total number of nodes searched by a factor of five on room graphs, and a factor of two on video game graphs. On maze graphs, the efficiency of the heuristic is not as high, reducing the total number of nodes searched by a factor of 1.9. Additionally, the execution times are even more noteworthy than the number of nodes expanded. Using the additive heuristic on room graphs resulted in a 29-fold reduction in execution time compared to using the Manhattan Distance heuristic, and on maze graphs it resulted in a 7.6-fold reduction. Using the additive heuristic on video game graphs resulted in a 2.5-fold reduction in execution time. While this is not the same as graph switching, it does show that manipulating graphs can yield significantly improved results when performing specific tasks.

IV. Human Roaming Modelling

The modelling phase involved having human subjects undertake a roaming experiment. So, generalised behaviours can be identified and incorporated into a model, which will aim to imitate an average human player roaming characteristics.

A. Data Capture Experiment

The data capture experiment was conducted by having subjects roam the map in search of eight coins. A heatmap was generated by adding a standard A* graph, each node was given a collider detection and when a subject intersected with the collider, a counter specific to that node was incremented by one. Constraints were added to prolong the overall length of the experiment, so a more accurate model of roaming could be achieved. Only one coin is present on the map at any given time. This was to prevent chaining where the subject spots a coin as they are moving to collect another coin. When a new coin spawns, it can spawn anywhere on the map, but not in view of the subject current position and cannot collide with terrain. This was to prevent the chance of coins repeatedly spawning close to subjects.

The purpose of this method was due to the separation of the navigation model and A* graph. Wherein, the NPCs uses the graph to plot a route, but it is not part of the overall navigation model.

Figure 1 shows an overview of the map, each number represents a room, additionally, there is a large open foyer area in the centre of the map.

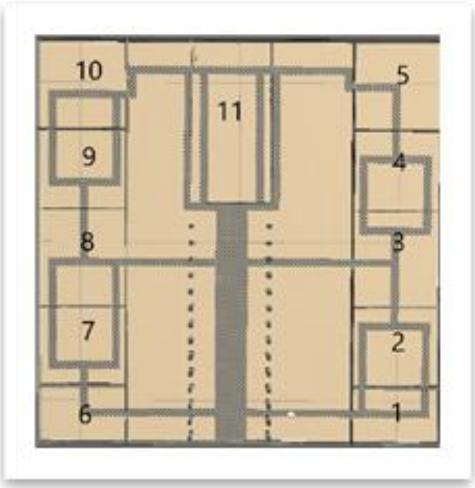


Figure 1. Map Overview

The map was specifically designed this way because it represents the type of map occasionally seen in a video game. This is where there are a number of small, confined

areas and then one open space from which the smaller areas are connected.

Non-model related data was captured so a comparison can be made as to the efficiency of model in relation to the overall performance of the roaming behaviour. This was to conclude if the act of roaming is random, or if there was a more significant strategy as to why subjects used certain doorways and routes. Therefore, the position and rotation of the subject was logged every 0.5 seconds, which can be input back into the experiment for behaviour observation by a researcher.

B. Results and Analysis

A total of 30 subjects took part in the experiment. Figure 2 shows the combined heatmap of all subjects. The result shows an interesting trend where subjects were more likely to traverse the outer edge of the map, which influenced which doorways were likely to be used.

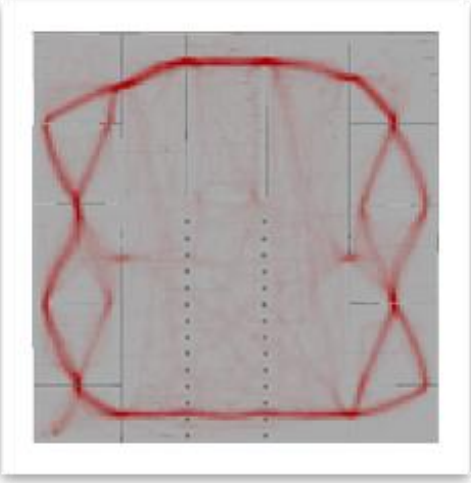


Figure 2. Heatmap and Room Numbers.

This demonstrates that roaming is very strategic, and subjects increase their likelihood of finding a coin by increasing their viewing coverage. It also highlights that roaming routes are funnelled via doorways, and it is likely that subjects' navigation decision-making was primarily limited to the space between doorways. The experiment confirmed initial results, which showed that human players had preferred routes through the map [3]. While this is not surprising as subjects need to use the doorways to traverse the environment, as the map resembles a typical office, it indicates the importance of map design and the strategic value of funnel points. Even in open world maps, generally there are points of interest, with routes, such as roads, leading directly to these areas. There is some heat in the centre of the foyer, this was caused by coins appearing in random locations including the middle of the foyer. This is evident by observing the heat patterns around the foyer doorways, which show a significant change in direction from the initial roaming path towards the location of the coin, indicating that as subjects are entering the foyer, they spot a coin, and then readjust pathing to collect it.

As roaming is strategic, map coverage is therefore an important objective. Figure 3 shows an example of the amount of map uncovered by a randomly selected subject. It shows that at the end of the experiment >95% of the map has been revealed, with a small area in the corner of room 4, which was not uncovered.

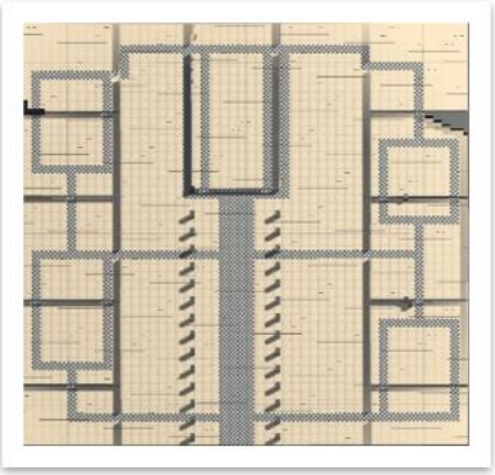


Figure 3. Map Coverage.

There were three key generalised characteristics:

- Player Positioning: Subjects were more likely to traverse the outer edge of the map, thus, increasing viewing angle to cover more map.
- Peeking: Subjects occasionally 'peeked' into the foyer area, this involved moving to the doorway connecting to the foyer for a quick look, before continuing their intended route.
- Rapid Room Scanning: Upon entering a room, subjects were likely to quickly scan the room as they continued to move towards the next doorway.

These behaviours were consistent across most subjects and emphasise that there is a clear logic behind roaming that is not a random undertaking. It is an organised activity where the objective is maximising the efficiency of map coverage.

A critical behaviour that emerged was the speed in which subjects' navigation behaviours changed when new information was presented. While the roaming was methodical, when subjects identify a coin, the behaviour shifts immediately to acquiring the coin. The behaviour changes from looking around the map, to a focused behaviour where the subject remained fixated on the coin and moved directly to retrieve it. Figure 4 shows the results from one subject. The circles show where the subject spotted a coin and immediately breaks away from the roaming route, then after collection they resume on the same roaming route. In one instance, the subject can be seen to traverse the width of the map in a near straight line when spotting a coin.

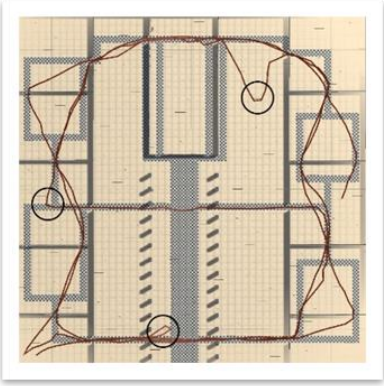


Figure 4. Beeline Behaviour.

This suggests that the navigation model could require a subset of models for the various behaviours associated with moving throughout the map. This could lead to establishing that navigation is more than point to point pathing, but an expression of behaviours related to fulfilling specific objectives. This could explain why subjects were scanning the surroundings when roaming and why they were fixated on their target when collecting coins.

When discussing the act of 'peeking', it is a common pattern among subjects. The data shows that most subjects would look towards doorways they are not moving towards. This gives them the opportunity to see the area behind the doorway, thus, allowing them to scan more of the map. However, at times subjects would actively move to a doorway, scan the area before moving back to their original route (Figure 5). To calculate the parts of the map the observed, a black overlay was added which is removed if inside the subject camera field of view. Ensuring this did not interfere with the experiment, the black overlay was culled from the camera, therefore, would not be rendered during the experiment.

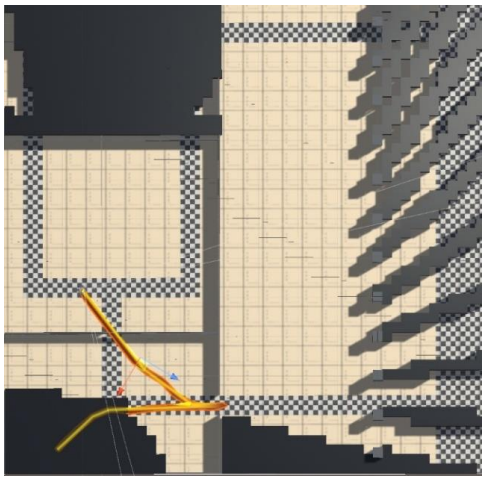


Figure 5. Peeking into Foyer

In this example the subject started in the bottom left and the yellow/red line shows the path they took. They move to peek into the foyer and visually scan a large area, after that they turn around and move towards the doorway adjoining the next room. The importance of this mean they were able to check a significant part of the foyer and a room with very little time lost.

V. Roaming Model Analysis

A roaming model analysis experiment was conducted to determine if the roaming model developed represented the characteristics of an average human subject. The objective was to compare the human data and NPCs directly, to establish the accuracy of the model and determine if there were any negative consequences from using a heatmap.

A. Pathfinding and Roaming Model

There is a distinction between pathfinding and roaming. Pathfinding uses the heatmap weighted A* algorithm to plot a route between two points on the map. Whereas the roaming model controls the navigation decision-making and behaviour of the NPC as it moves between these two points.

The technique uses the heatmap to adjust individual node cost in the A* graph. This was achieved by adding all the specific node counters from the data capture experiment, then subtracting this weight from individual nodes when creating the A* graph. Figure 6 is a pseudo code example of the method used to create the A* graph with heatmap.

```
Loop X grid size
   Loop Y grid size
      create node world position
      check if node is walkable
         int movement penalty = 100
         walkable = True
         movement penalty -= heat weight value
      Else
         walkable = False
      Add node to array
```

Figure 6. Pseudo Code for Heatmap A* Graph.

As the distribution of cost between neighbouring nodes can vary significantly. It was decided that a smoothing technique was required to blur the differences. This was also required to help prevent NPCs occasionally traversing very close to walls as nodes neighbouring non-walkable nodes, such as walls, had their cost increased.

The smoothing technique used a box blur algorithm to normalise the cost of a node. A compromise was made where the box blur was set to 3x3, because when testing 2x2 the blur was not enough and when using 4x4 and 5x5, the smoothing was so significant that the heatmap had no effect. Figure 7 displays the box blur equation. Each number represents the weight cost of a node, the centre number is the

node being blurred by adding all weights then dividing by the number of neighbouring nodes.

$$\frac{1}{9}\begin{bmatrix}1,1,1\\1,1,1\\1,1,1\end{bmatrix}$$

Figure 7. Box Blur Equation

Subjects showed that doorways provided pivotal and strategic points on the map, as they are funnel points and are the only means of traversing between rooms. Therefore, as NPCs were restricted to information only in view, doorways became a central point to the model. Each NPC stored personal data about doorways and assigned a dynamic weight cost to each doorway, which reflected the heat observed from the subject experiment. The NPC will attempt to prioritise the doorway in view with the highest weight value. When successfully using the door, it will temporarily decrease the weight to prevent room cycling. Additionally, when NPCs have selected a doorway, the destination point is not a fixed point, but a random location slightly beyond the destination. The purpose of this was to reduce patterns forming when reaching the destination and calculating a new path. This is a common problem with some NPCs as they will perform the same action, at the same time, which can become noticeable by an external observer.

Unpredictability is an element of human players gameplay in an FPS game. There is a probability of performing a certain action in a scenario, but it is never certain. This was reflected in the roaming model, which aims to reduce predictability, but remain logical and consistent with human behaviour. This was achieved by implementing a random number generator of between one and ten, which represented the probabilistic outcome.

When entering a room, NPCs had an 80% chance to scan the room as they moved to the next location, as well as apply special attention to looking at other doorways. Peeking had a special importance when roaming because subjects used this technique to tactically scan open spaces without entering the area.

Lastly, when traversing open spaces, human subjects showed an awareness of their surrounds and took advantage by occasionally looking towards the open spaces, while still moving towards their intended location. This gave the appearance of the subject strafing as they were not moving in a forward-facing direction. This was modelled by enabling the NPC to have awareness of the distance between itself and open spaces to their left and right. Using this distance and a probabilistic algorithm, the model decided whether the NPC should scan the left or right side. After the NPC has successfully scanned the environment, a timer is started to ensure that the NPC does not keep repeating this action in a short space of time, a behaviour not seen in human subjects.

B. Experimental Protocol

The purpose of this experiment was to directly compare human subjects and NPCs. It was decided that having NPCs undertake the same experiment as the data capture experiment would provide a good basis to compare the results.

To remain consistent, NPCs run the experiment several times, varying the weight impact of the heatmap, so it could be determined which weight better represented the characteristics of the average subject. There were four different weight profiles. The heavier the weight profile, the higher the base node cost on the graph, which is represented by the darker the colour (Figure 8).

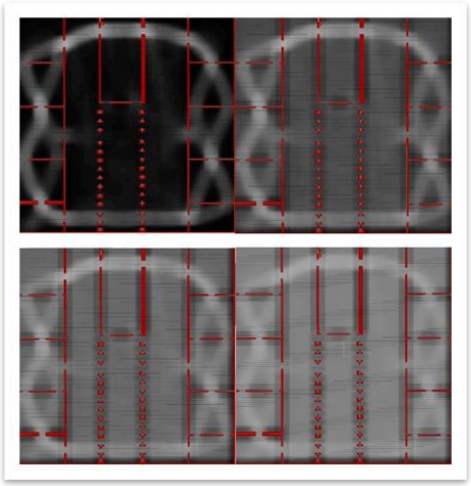


Figure 8. Heatmaps Comparison.

It shows the significant difference between weight costs of the different profiles. The top left profile aims to aggressively influence the NPC to adhere to the heatmap. While the bottom right profile was aimed at being more of a light influence on the pathfinding. This is a promising sign because it means the heatmap is working as intended and the degree of change between the profiles demonstrates that the model should be flexible in its application. This presents a novel approach to pathfinding as the heatmap is not strictly limited to player data. It could be used to prevent NPCs roaming the same areas by increasing the weight cost of nodes based on its own heatmap which is calculated over a set length of time. This technique could also make use of multiple graphs, each with a unique heatmap. It would enable developers to have more control over the navigation behaviours of NPC's, based on the task they are undertaking.

Figure 9 shows an example of the A* graph without the heatmap. While this profile was only used once in this experiment, it is a good comparison to show the influence the heatmap has on the A* graph.

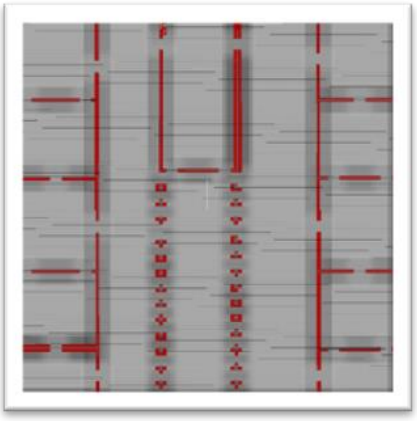


Figure 9. No Heatmap Example.

The node cost smoothing technique was added to all the graphs. This ensured that neighbouring nodes did not have wildly different movement costs. This was essential because in a heatmap, some neighbouring nodes could have significant variation in cost, which would result in the NPC having a jerking motion as they moved.

While some NPCs were required to collect eight coins for a direct comparison with individual human subjects, other NPCs were required to collect forty coins per run, which was the equivalent of five subjects. This number was decided as the purpose of the experiment was to analyse the generalised roaming route of the NPCs. A line renderer was used to track NPCs routing. Therefore, a compromise was required where it would provide enough data to make conclusions, but not too many where the lines become saturated and confusing.

As with the the human subject experiment, the whole map is covered with a black fog, which is instantly removed when entering the view of the NPC. The fog provided a measure of the areas of the map the NPC has scanned and allowed comparison with observations of the human subjects.

C. Results and Analysis

When directly comparing the four weight profiles, the results show that amplifying the significance of the heatmap on the cost of the A* nodes, NPCs pathfinding was noticeably affected. Generally, the model has a positive effect on the navigation and each of the heatmaps accurately reflects the roaming patterns observed in the human subject experiment (Figure 10). However, it also had an adverse effect when not roaming. NPCs were taking very inefficient routes to reach a specified location, such as moving to a coin location. On some occasions, NPCs were not making a beeline behaviour after identifying a coin. They would lose sight of the coin and move through multiple rooms, before finally acquiring the coin. Although this behaviour is clearly at odds with that exhibited by the human subjects it does comply with the lowest cost path calculated by the modified A* Algorithm.

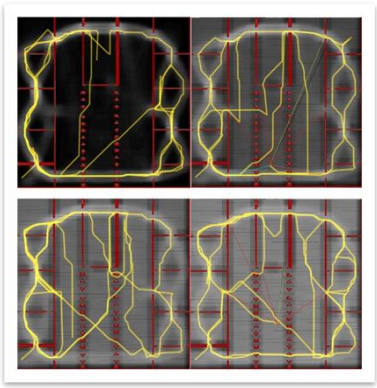


Figure 10. Heatmaps Analysis.

An important observation is in the centre of the map because there is a correlation between the base node cost and likelihood of cutting across the map to reach a destination when roaming. This opens the pathfinding to a degree of flexibility because multiple graphs could be created and the model decides which to use, or if the graph is regularly updated, it can decide how aggressive the roaming should be in relation to the heatmap. This could be useful in a scenario where NPCs are tasked with tracking players and the developer does not want to use scripting to force interactions. Similar research has been undertaken where NPCs are influenced by pheromones, which are generated by other game agents with positive results [13]-[15]. While these examples are generally focused on real-time strategy games, and are intended to explore swarm intelligence, commonality can be derived with the technique presented in this paper. Subject to further investigation, research could be undertaken where the players emit pheromones that temporarily decreases the cost of nodes within the vicinity.

Analysing the model when the coin count was set to forty, the results remain consistent with what was observed with eight coins (Figure 11). In this example, a moderately aggressive base node cost was chosen to help prevent NPCs using the middle of the map to roam, but not too costly that NPCs would take inefficient and irregular route when moving to a coin. The results show the heatmap has a very strong influence on roaming, and when moving to a coin, the NPC would use the middle of the map.

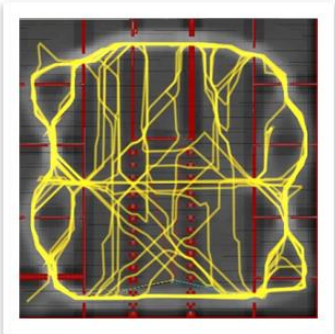


Figure 11. Forty Coin Heatmap Analysis.

When comparing the heatmap against a normal A* graph, the results appear somewhat similar, however, when scrutinising the straightness of the paths, it shows a degree of difference. Figure 12 shows the heatmap A* (left) and the standard A* (right). The heatmap lines show they are not straight, but instead have a slight meandering characteristic.

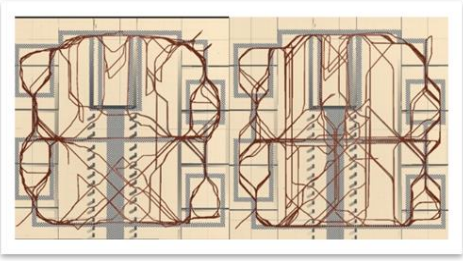


Figure 12. Heatmap A* vs Standard A* Comparison.

Zooming in to specific areas further highlights intricate differences between the heatmap and non-heatmap A*. Figure 13 focuses on a single room. The right image shows a uniform pattern, whereas the left image is less structured that is more reminiscent of human subjects. The heatmap NPC (left) shows that by implementing the random destination point, it affects the pathing where the difference is subtle but clearly different.

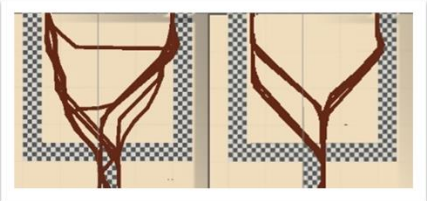


Figure 13. Room Comparison.

When compared to a human subject in a similar room (Figure 14), it shows NPCs using the heatmap is more akin to the subject than the NPC using the standard A* graph. While this is a subtle observation, it is nevertheless important, because when NPCs move in straight lines and display the same movements, these patterns cannot become noticeable when viewed over a length of time.

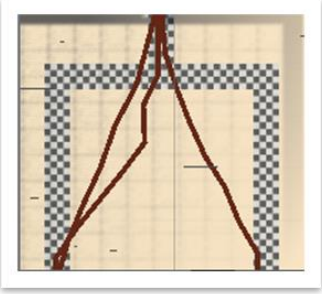


Figure 14. Human Subject Room Analysis.

A key feature of the roaming model was imitating how human subjects entered a room and the time it takes to start moving to a new location. Figure 15 shows the doorway exit and entrance trajectory. The results indicate that the model is working as intended. There are no identical paths, all adhere to a logical tactic, but there was little pathing efficiency cost.

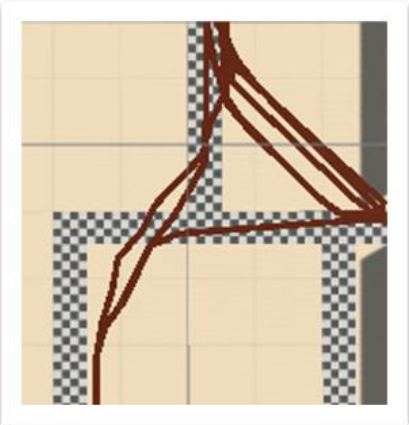


Figure 15. Door Trajectory Analysis.

A key strategy used by human subjects was peeking into large areas. The effect of peeking can be seen most clearly when looking at the fog, the red line indicates the path the NPC followed (Figure 16). It moved into the room, peeked into the foyer area before resuming initial route. Such characteristics are integral to having an accurate imitating roaming model because it projects a degree of intelligence when observed by a player.

Figure 16. NPC Peeking Identification.

When discussing the map view coverage, with the objective set to collect eight coins. NPCs showed approximately the same level of map coverage as human subjects was achieved (Figure 17). This shows that the

roaming model can comfortably seek out and acquire anything within the realms of the A* graph.

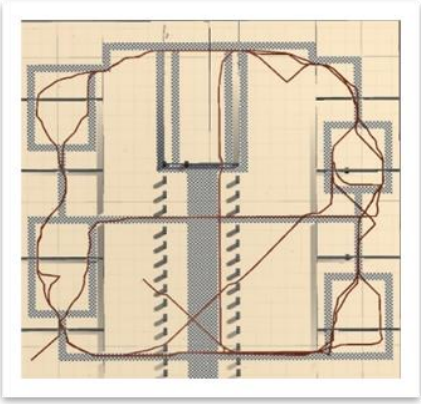


Figure 17. Roaming Model Map Coverage.

Being able to fully scan the environment is a crucial aspect of navigation. When the goal location is not predetermined, if the NPC cannot scan and analyse the entire map, some goals might become impossible to complete.

Finally, when analysing performance, a good measurement is the completion time, as it refers to the efficiency of the navigation. Table 1 shows the time to complete comparison between human subjects and NPCs using various weighted heatmap A*.

Table 1: Completion Time Comparison

Heatmap Weight	Time (Minutes)
0	6.3
25	4.66
50	7.7
100	5.9
Human Subject	6.6

While the results vary slightly, this could be partly due to the randomness of the coin spawns. What the data does indicate is that the model is approximately as efficient as human subjects. This is encouraging as the heatmap is influencing the navigation behaviour of the NPCs and also the efficiency, which are two important areas of gameplay.

VI. Graph Switching

Graph switching is the term being used for using multiple graphs in A* algorithm. Typically, A* has one graph, and based on cost and heuristics, it finds an optimal path between two points. This paper intends to build upon this methodology and introduce multiple graphs, these graphs have a specific purpose and will be used depending on the task being undertaken. This experiment will use two graphs:

- Roaming: This will be the heatmap generated graph and will be used only when roaming.
- Standard: This graph is a standard uniform graph, with a smoothing technique to stop NPCs walking next to walls.

While it is possible to have more than two graphs, and ideally, graphs would be modelled based on the behaviour being displayed. For the purposes of this paper, it was important to directly compare the heatmap graph against a normal graph and not over complicate the research.

A. Graph Switching Analysis Experiment

This experiment involves two deathmatches, in which a subject will be tasked with competing against two NPCs, in an all verses all scenario. The first deathmatch NPCs will use a standard A* graph only, while the second deathmatch, NPCs will be using the graph switching model.

The objective is to achieve eight eliminations before any of the NPCs. All participants, both subject and NPCs, have access to the same weapons and have the same amount of health. The character models are consistent across all characters with no animations. This was decided as poorly implemented animations could affect the perception of the NPCs. All characters use the same movement, Table 2.

Table 2: Character Controls

Movement	Description
Forward	Moves in the direction the player is facing
Back Peddle	Slowly moves in a backwards direction
Strafe Left	Side steps to the left
Strafe Right	Side steps to the right
Look (Aim)	Rotates the camera and weapon, with crosshairs fixed in the center of the screen

The ability to jump was removed as there are no pitfalls. Eliminations are awarded only to the player that got the last hit. Therefore, if two players are attacking the same target, only one will be accredited with the elimination.

The shooting mechanics was the same for NPC and subject. All weapons had the ability to be shot off the hip or aiming down sights. Each weapon had unique characteristics, such as damage, attack speed and bullet spread. There were three weapons:

- Pistol: Slow fire rate, high damage, and high kickback
- Assault Rifle: High fire rate, medium damage, and low kickback
- Shotgun: Slow fire rate, wide bullet spread and high kickback

These weapons were used because they are frequent in modern video games, which should help subjects quickly familiarise with the experiment when they start.

Throughout the map there are six collectable items, three of which were medic packs, which award fifty health points, and three ammo packs that award one clip of ammunition to the currently equipped weapon. The map layout is the same

as the previous experiments, this was to help keep consistency in the analysis when comparing data.

Both NPCs will be using a standard combat model, and a finite-state machine (FSM) to transition between gameplay states. The graph switching occurs when the NPC changes states and updates the navigation model to instruct which graph to use should a path be requested. This enables the model to explicitly state the navigational behaviour it needs to project, as the FSM can only occupy one state at a time.

During development a feature was developed to resemble a 'bug' in the code. This provided a small chance that NPCs could become stuck and not move. The purpose for this was to see if subjects noticed the bug, and if so, what affect it had based on the feedback.

As this is a comparison experiment, the data required will be directly from subjects. After the deathmatch concludes, subjects are asked to complete a questionnaire, which is related to their experience in the deathmatch. This paper is a generalised look at player navigation, it was decided that there would be no subject requirement, and anyone that was interested in partaking could complete the experiment once. The experiment files were posted online so the subject could complete it using familiar peripherals and under normal conditions, as though they were playing a video game. The experiment was conducted in English, and subjects remained anonymous, with no personal data required or acquired.

B. *Results*

When comparing and analysing the feedback, the overall results were promising and showed the graph switching working as intended. It also illustrates that subjects identified the irregular behaviour caused by the bug and it had a detrimental impact on how they perceived the NPCs.

The overall impact on perception can be observed when analysing overall navigation feedback because this is when subjects directly interact with the NPC or observes from a distance. Figure 18 displays the specific feedback from the non-modelled NPC (left chart) and graph switching NPCs (right chart).

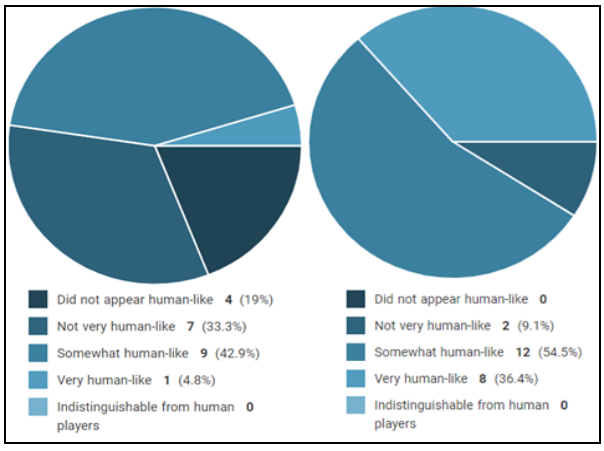


Figure 18. Subject Feedback Comparison

The data indicates that roughly 90% of the subjects considered the NPCs with graph switching to appear fairly human-like. While more than 50% of the standard A* NPCs were poorly received. It can be assumed that this improved perception in the graph switching NPCs is directly related to how they appeared when roaming, because the A* graph for combat was the same as the non-modelled NPCs.

When analysing the data, it is important to focus on areas where the perception was at its weakest, so these areas can be addressed, a good metric for this was directly asking the subject for feedback as to where the model failed.

Figure 19 displays a portion of the feedback from the subjects that answered the question relating to the graph switching NPCs only.

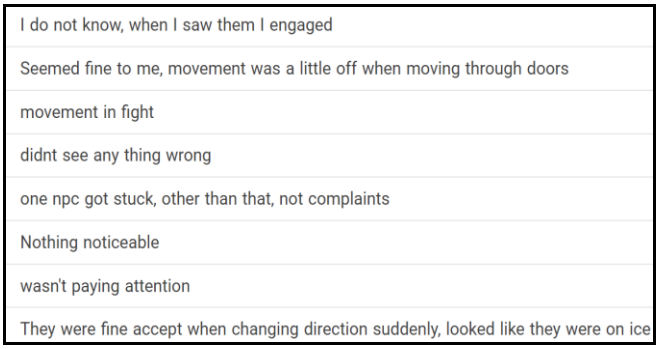


Figure 19. Specific Navigation Feedback

The results appear quite mixed with some not directly addressing the weaknesses of the model, however, some offer valuable insight. For instance, the moving in combat has yet to be modelled and this was indicated to have broken immersion. It should also be noted that some comments are confusing movement to animation, so the suggestion that NPCs moved like they were on ice, if the character was controlled by a human, the appearance would be the same.

The feedback shows that the 'bug' did occur and some of the subjects witnessed this and has reported it as the least believable part of the navigation modelling. This is a crucial discovery, because it shows the potential impact that a minor bug can have on the perception of the NPC, and that believability needs to be a continuous and when it fails, regaining the illusion can be difficult.

Figure 20 highlights a section of the feedback when asked which part of the NPCs was the least believable in relation to human player gameplay.

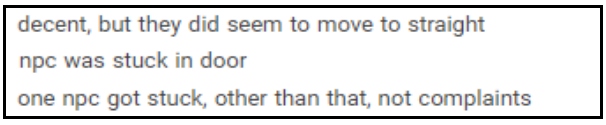


Figure 20. Feedback Specifying Bugs

This highlights just how significant bugs can influence perception and break immersion, and while it may be a minor issue, the impact can be lasting. Furthermore, it illustrates that subjects have a general level of awareness, and when this awareness detects something not quite right, it can also influence opinions.

Finally, when categorising navigation into three parts and asking subjects if any of these areas were weak. Figure 21 shows that only 36% of the subjects submitted a weakness in the navigation.

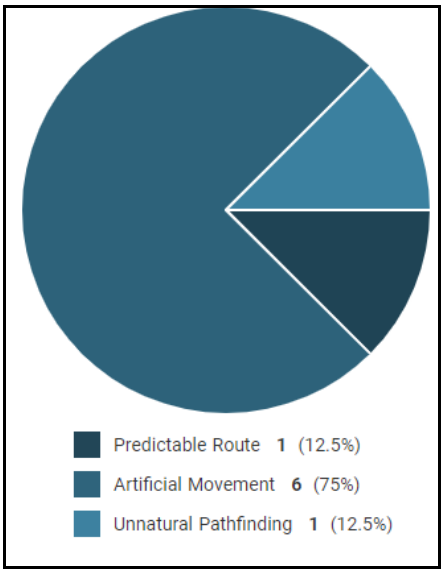


Figure 21. Model Weakness Feedback

Most of the feedback indicate that 'artificial movement' was the least believable, and when compared with some of the comments from the open-end question about navigation (Figure 19), it implies that the break in immersion could be partly from the intentional bug add to the experiment.

Graph switching represents a step forward in imitating human navigation behaviour, it showed that a multi-model strategy is required to imitate the various behaviours associated with the multitudes of navigational purposes. While for this experiment it proved a good solution and encapsulating more behaviours should improve the perception of the NPCs and yield even better results. However, eventually, this methodology could result in exponential complexity, if too many navigational behaviours are required. It could therefore be argued that a better approach moving forward would be to tie individual navigation behaviour models to the decision-making modelling, this would give the decision-making modelling direct control over the navigation and instruct it which subset of models it needs to be using at a given time. If used with Finite-State Machine (FSM) or Goal-Orientated Action Planner (GOAP), a graph could be modelled or linked to each goal or state.

VII. Conclusion

The objective of this paper was to explore if human player navigation data can be used to create a heatmap, for the purpose of adjusting the weight cost in an A* graph to influence the pathfinding.

The roaming model demonstrates that using a method, which restricts pathing decisions based on what it can see, and using a heatmap weighted A* algorithm, a good imitation of the general roaming behaviours of a human player can be achieved. This hybrid model was able to take advantage of tags in the environment so the NPC could make decisions based on what was in view. Thus, removing omniscient characteristics often associated with NPCs and which can be clearly identified by players. The heatmap weighted A* offers a unique approach to influencing pathfinding so that NPCs use frequently travelled areas, making the NPC interactions more natural, than scripted interactions that can appear forced.

However, it was clear that using a designated A* roaming graph had negative implications when used for other navigation tasks. Therefore, the practical application of the model would need to incorporate a multiple graph solution, in which the A* would be applied to an appropriate graph, based on the task the NPC is undertaking.

The graph switching indicated that it was a viable solution for changing the navigation behaviour of NPCs, depending on the type of task they were undertaking. While more research will be required to evaluate the full usefulness of this technique, modelling specific navigational models could enable NPCs to have more depth to their gameplay. Furthermore, it will allow developers to have greater control over how the NPC should be perceived and when modelled on human-based data, could become more believable.

References

[1] P. Williamson and C. Tubb, "Heatmap Weighted A* Algorithm for NPC Pathfinding," In SIMUL 2022: The Fourteenth International Conference on Advances in System Simulation, pp. 15-21, 2022.

[2] I. Millington and J. Funge, "Artificial intelligence for games," CRC Press. 2018.

[3] P. Williamson, "Modelling Human Behaviour in a Virtual Environment," [Unpublished Masters Thesis]. University of South Wales, 2019.

[4] Z.A. Algfoor, M.S. Sunar and H. Kolivand, "A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games," International Journal of Computer Games Technology. 2015.

[5] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah and E. Gunawan, "A systematic literature review of A* pathfinding," Procedia Computer Science, pp. 507-514, 2021.

[6] W. Ziqiang, H. Xiaoguang and L. Xiaoxiao, "Overview of Global Path Planning Algorithms for Mobile Robots," Comput. Sci., vol. 48, pp. 1-16, 2021.

[7] S. Permana, K. Bintoro, B. Arifitama and A. Syahputra. "Comparative analysis of pathfinding algorithms a*, dijkstra, and bfs on maze runner game," IJISTECH International J. Inf. Syst. Technol., vol. 1, no. 2, pp. 1, 2018.

[8] Y. Sazaki, H. Satria and M. Syahroyni. “Comparison of A∗ and dynamic pathfinding algorithm with dynamic pathfinding algorithm for NPC on car racing game,” 11th International Conference on Telecommunication Systems Services and Applications, pp. 1-6, 2017.

[9] I. Makarov, P. Polyakov and R. Karpichev, “Voronoi-based Path Planning based on Visibility and Kill/Death Ratio Tactical Component,” In AIST, pp. 129-140, 2018.

[10] N. R. Sturtevant, D. Sigurdson, B. Taylor and T. Gibson, “Pathfinding and abstraction with dynamic terrain costs,” In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 15, no. 1, pp. 80-86, 2019.

[11] H. Pan, “Pathfinding and Map Feature Learning in RTS Games with Partial Observability,” In AIIDE Workshops 2021.

[12] K. Anderson, “Additive heuristic for four-connected gridworlds,” in Proceedings of the 3rd Annual Symposium on Combinatorial Search, pp. 10-15, 2010.

[13] D. Daylamani-Zad, L. B. Graham and I. T. Paraskevopoulos, “September. Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games,” 9th International Conference on Virtual Worlds and Games for Serious Applications, pp. 39-46, 2017.

[14] X. Chen, et al., “Towards believable resource gathering behaviours in real-time strategy games with a memetic ant colony system,” Procedia Computer Science, vol. 24, pp. 143-151, 2013.

[15] A. Rafiq, T. Kadir and S. N. Ihsan, “Pathfinding algorithms in game development,” In IOP Conference Series: Materials Science and Engineering. vol. 769, no. 1, pp. 1-12, 2020.

Active and Cooperative Learning in a Multicultural Software Engineering Class: Impact of Switching from Offline to Online Classroom Modality

Simona Vasilache
Faculty of Engineering, Information and Systems
University of Tsukuba
Tsukuba, Japan
e-mail: simona@cs.tsukuba.ac.jp

Abstract - **Three years after the start of the Covid-19 pandemic, the academic world is still feeling the effects of the sudden changes brought along by this unprecedented event. With most academic institutions abruptly switching to online teaching in 2020, pursuing the goals of implementing active and collaborative learning has turned into a challenging endeavour. New strategies had to be imagined and the question arises whether this abrupt transition to online teaching can remain successful, especially in multicultural groups. This paper is based on the experience of teaching a graduate software engineering course to a multicultural group of students. It describes the active and collaborative learning strategies employed during offline classes and highlights the impact of switching from an offline to an online class modality. The results show that the sudden switch to online teaching was managed successfully and the active and collaborating teaching activities were successful in fulfilling the course objectives and offering student satisfaction with the course and the learning process.**

Keywords - software engineering; active learning; cooperative learning; multicultural environments; classroom modality.

I. INTRODUCTION

Active and collaborative learning are two important concepts with a multitude of benefits in the education field, being applied in teaching various disciplines. Software engineering is one such discipline, in which a hands-on approach is traditionally considered important ([1], [2]). Maxim et al. noticed that students may be exposed to the necessary concepts during courses, but they are often "not asked to apply these skills in project settings" [3]. They showed that active learning can improve software engineering education. By implementing active learning, students can acquire knowledge in a practical way; by using collaborative learning, teachers can create an environment emulating future work environments, where software engineers need to work together to produce software applications.

At the same time, multicultural environments are ubiquitous – in the workplace, in the academic world, in our daily lives. Teaching in multicultural environments has its own set of challenges [4], but also its particular advantages [5]. In case of teaching software engineering, a multicultural classroom offers the benefit of simulating distributed teams of software developers. Students learn in one classroom together, but they can get a glimpse of how they would have to interact with their future developer colleagues belonging to different cultures and thus having different values, different behaviour and various working styles [5].

Software engineering education suffered, along with many other disciplines, during the Covid-19 pandemic. Academic institutions were forced to suddenly move their courses to an online format, with very little preparation. The changes in classroom modality, from offline to online, created new challenges for instructors and students alike. Academic institutions and the process of instruction faced an unprecedented situation; going through it provided numerous valuable lessons for everyone involved in the learning process. In our previous paper [1], we highlighted some of the challenges encountered, along with lessons learned in implementing active learning while teaching software engineering to a multicultural group of graduate students at a national university in Japan, during an introductory level software engineering course. This paper extends our work with data from the course held in 2022 (the third year of online teaching), as well as with more considerations regarding collaborative learning. Moreover, it discusses to what extent the various factors which make a successful course depend on the class modality, i.e., face-to-face or online.

The structure of this paper is as follows. Section II describes the background of our work, along with related work. Section III provides a description of our course, whereas Section IV illustrates the implementation of active and collaborative learning. Section V includes a discussion on course effectiveness and lessons learned during our study; conclusions and directions for future work are provided in Section VI.

II. BACKGROUND

This section will offer an overview of concepts like active learning and collaborative learning, and it will present recent related work, published in the context of the sudden switch to online learning that took place in 2020 after the onset of the Covid-19 pandemic.

A. Active Learning and Collaborative Learning

A lot has been written about the concepts of active learning and cooperative learning and their benefits. Simply explained, active learning is “any teaching method that gets students actively involved; cooperative learning is one variety of active

learning which structures students into groups with defined roles for each student and a task for the group to accomplish" [6]. During traditional, lecture-based, "passive" classes, students listen to experts who impart their knowledge [7]. In 1996, Gremmels used a truck analogy for lecture-based teaching: "So we in effect load our pedagogical dump truck as full as we can, back it up to the classroom, and unload it onto our students, burying them in teaching ... When we use the dump truck method, we overwhelm our students with more skills and strategies than they can possibly absorb in an hour. That's our first mistake. Then we fail to give students the opportunity to practice any of the strategies and skills, virtually guaranteeing that they won't be internalized." [8]. During active learning, students must do more than just listen: "they must read, write, discuss, or be engaged in solving problems" [9]; importantly, they must engage in "higher-order thinking tasks as analysis, synthesis, and evaluation" [9].

Based on a meta-analysis of 225 studies, Freeman et al. concluded that active learning increases student performance in science, engineering, and mathematics [10]. This was based on reported data on examination scores or failure rates when comparing student performance in traditional and active learning [10]. Generally, researchers agree that projects using "active methodologies" help students to "develop deeper knowledge and apply it in a practical way according to a work plan" [11].

Cooperative learning is one form of active learning, in which students work together with the goal of maximizing their own learning, as well as their peers' learning [12]. According to Keyser [6], planning is important for cooperative learning to work well. It is essential to clarify the role of each member of the group, as well as to carefully consider the size of the group [6].

Both active learning and cooperative learning promote the achievement of student engagement, which is a common goal of all educators in academic institutions.

B. Emergency Remote Teaching

The 2020 Covid-19 pandemic forced many academic institutions to move their courses to an online format. Online and remote learning became a necessity in times of lock downs and social distancing [13]. The sudden shift to an online format, taking place with virtually no preparation, was termed Emergency Remote Teaching (ERT) [14]. As its name suggests, this way of teaching was different from the well-established online class teaching method, which took years of work and gradual improvements. Its purpose - as a response to crisis - was to provide temporary access to instruction quickly and reliably [14].

Unfortunately, most faculty had no prior training in teaching at a distance, and most universities were unable to support them in the way universities with traditional online programs generally do [15]. The courses which were to be delivered in this manner were meant to be a temporary solution to an unprecedented, emergency situation, and not a long-term replacement of face-to-face courses. In reality, ERT continued for a couple of years in many academic institutions; a third year on remote teaching should not be named "emergent" anymore. In fact, Stewart at al. use the term "sustained remote teaching" (SRT) to represent the method of teaching employed in the second year of the pandemic [15]. Overall, the abrupt change in delivering courses in 2020 brought innumerable challenges for instructors and students alike.

C. Related Work

After the onset of the 2020 Covid-19 pandemic, as mentioned earlier, most higher education institutions had to suddenly switch to an online learning. The experience and the lessons learned were described in various research papers. In 2020 alone, Google Scholar lists a little over 2000 research paper mentioning "emergency remote teaching". Later on, as of June 1st 2023, more than 20,000 papers dealing with this subject appear on this platform.

One of the earliest papers published after the proposal of the "emergency remote teaching" term [14] was that of Shim and Lee [16]. Using thematic analysis, they analyzed South Korean college students' experiences of ERT within the first 4 weeks of ERT, in March-April 2020. Positive features, along with areas of complaints in the context of ERT were highlighted. Their conclusions emphasized the importance of college students' educational environment, and showed that "the quality of interactions can vary depending on the teachers and technology used" [16]. In the same year (2020), Bozkurt and Sharma [17] acknowledged that "online distance education is one thing and emergency remote teaching is another thing". Moreover, they remarked that the distinction between these two is particularly important, because the degree to which educators believe in distance education during the early days of the pandemic would "play a significant role in the prosperity of distance education in a post-COVID world" [17].

The experience of dealing with ERT was the subject of numerous research works after 2020. In [18], Chierichetti and Backer explored faculty perspectives on teaching engineering during ERT; their view is that faculty members in engineering "have always viewed online teaching with skepticism". Their study concluded that the sudden transition to online teaching in engineering at the public university where they teach was mostly positive and that the teachers' concerns were mainly focused on methods of assessment and student engagement. Douglas et al. [19] provided recommendations for future moves to ERT, which included providing mechanisms for informal conversations students-students and students-instructors, availability of instructors for just-in-time feedback and questions etc.

Furthermore, Chiu's research [20] argued that the delivery of knowledge in a digitalized form is dynamic and it relies on the efforts of all course participants. The interactions between students and teachers were studied and the results pointed to the fact that the effectiveness of online learning is strongly reliant on the dynamic between the teachers and the students.

III. COURSE DESCRIPTION

This section will describe the course that constitutes the subject of this paper, its setting and content, as well as the number of students and format changes throughout the years.

A. Course Content and Description

This paper is based on the experience of teaching an introductory software engineering course named "Principles of Software Engineering". The course is offered as an elective for master's students in the computer science department at the University of Tsukuba in Japan, and it belongs to the "Degree Programs in Science and Information Engineering (Specialized Subjects)" category. The language of instruction is English, and the course takes place during modules A and B of the spring semester (each module is made up of 5 weeks, thus stretching the course over 10 weeks); upon successful completion, students acquire two credits.

The goal of this course is to introduce basic software engineering principles; students learn about the necessity of software engineering as a modern engineering discipline. The main topics covered include software development models, life cycle, agile methods, requirements engineering, user interface design, testing (verification and validation), project planning and management, software engineering tools (IDEs, UML) and business aspects of software development.

The participating students are a mixture of Japanese students and international students, with the latter group making up the majority. The reason behind a high percentage of international students is most likely related to the language of instruction. On one hand, the number of courses offered in English is not very large; international students take every opportunity to attend classes held in English. On the other hand, Japanese students often lack the self-confidence to enroll in classes not held in their native language.

Whereas most of the enrolled students belong to the computer science department, other participants major in material sciences, library, information and media studies, intelligent and mechanical interaction systems, policy and planning sciences etc. Occasionally, one or two undergraduate students enroll in the course, as well (as a general rule, only high achieving undergraduate students can attend graduate school courses). Also occasionally, exchange students take part in classes; they study mostly engineering, but they might major in other fields (e.g., linguistics).

Table I shows the total number of students and the number of international students participating in each of the 7 years since the course was established.

TABLE I. INTERNATIONAL STUDENT ENROLLMENT IN SOFTWARE ENGINEERING COURSE

	Total number of students	*Number of international students*	*Percentage of international students*
2016	15	9	60%
2017	26	18	69.2%
2018	35	24	68.5%
2019	66	33	50%
2020	34	28	82.3%
2021	53	37	69.8%
2022	66	33	50%

As can be observed, whereas the total number of participating students decreased suddenly, the percentage of international students did not reduce during the pandemic, despite the fact that there were simply fewer international students physically present on the university campus. Japan was one of the countries with the strictest border controls during Covid-19. Candidates from abroad already accepted in various undergraduate or graduate programs were unable to acquire a student visa for more than one year and thus unable to enter Japan. However, they were able to enroll in classes remotely, which was also the case for many local students, who chose to remain in their hometowns and be physically away from the university campus. Moreover, there were no exchange students entering Japan for the first year; in 2021, in theory, it was possible to become an "online" exchange student (remaining in own home country but allowed to enroll in the online classes offered by the partner university, i.e., the University of Tsukuba). However, this arrangement did not prove particularly popular and, consequently, there was a very low number of students participating in these exchange programs in the first two years after the onset of the pandemic. Fortunately, the situation has returned to pre-pandemic conditions in 2022, i.e., exchange students are now allowed to enter Japan and they can join the university for a period between three months and one year. Summarizing, whereas the ratio of Japanese to international students did not change much from the previous years, the total number of participating students was drastically reduced at first, but it slowly returned to its pre-pandemic numbers.

B. Format and Number Changes

The introductory software engineering course was first offered by the author in 2016; its establishment emerged from the necessity of providing more graduate school courses in English. Throughout the past 7 years, the course suffered a few changes, in terms of format and number of students. Figure 1 illustrates these changes, along with a brief overview of the style and type of activities used during classes.

In its first year, the course participants were evaluated based on a final exam; in subsequent years, students had to submit one mid-term and one final report, bringing the evaluation method in line with most of the other courses that the participating students enrolled in as part of their graduate studies in our university. Importantly, the students were allowed to submit their reports either in English or in Japanese, to accommodate the often-encountered lack of confidence in their English skills of the Japanese students.

Between 2016 and 2019, the course was provided in the traditional face-to-face format. It started with 17 students enrolled in its first year, followed by 26, 34 and 66 students enrolled in subsequent years, respectively. A major change occurred in 2020, with the sudden change to online format (i.e., ERT [14]), due to the Covid-19 pandemic. The number abruptly decreased to 35 students in 2020, followed by an increase to 53 students in 2021. By 2022, even though the class was still held online, the number of participants grew back to pre-pandemic levels (66 participants).

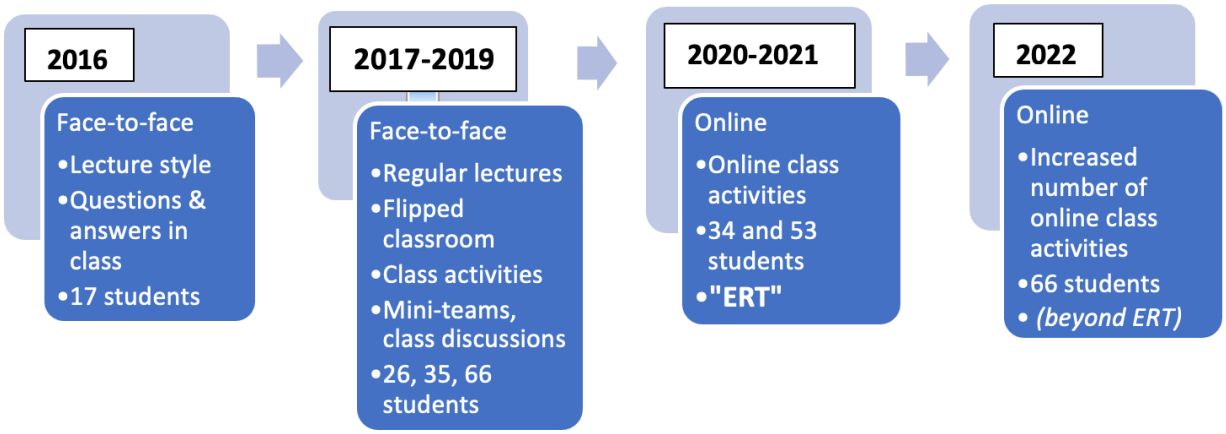


Figure 1. Software engineering course format and number of participants over the years

IV. Implementing Active and Collaborative Learning

This section will describe how active and collaborative learning were implemented throughout the 7 years that constitute the subject of this paper.

Note: the questionnaires mentioned in this section were administered as per the ethics regulations of the University of Tsukuba; approval was received for gathering and using the data acquired through these questionnaires.

A. Face-to-Face Format

a) Year 2016

The traditional face-to-face format was used between 2016 and 2019. In its first edition (in 2016), the course was taught in a rather small classroom, with - for the most part - the instructor lecturing in front of 15 students. Interactions between the instructor and the students took the simple form of asking students to answer various questions in class. The instructor was constantly hoping that these questions would elicit (lively) discussions among students. This happened increasingly more often, although with a very slow start. The first class introduced the students to the teaching style of the instructor (students may have been surprised at first by the requirement to participate, at least in giving brief answers). Using a small classroom facilitated the interactions: eye contact was easily established, and students found it difficult to engage in unrelated activities (e.g., checking their phones or their personal computers). Moreover, it was easy to address students directly, to remember and to be aware of the precise level of participation of each student.

Out of the 15 participants, 9 were international students; they were more eager to answer questions and generally more inclined to engage with the teacher and with their peers during classes. The instructor made sustained efforts to involve the Japanese students, as well, with relative success. As a general observation, it is worth mentioning that, most often, lectures in Japanese classrooms do not involve active participation of students. Traditionally, the instructor teaches new topics in the form of lectures, while standing in front of the classroom, whereas students simply listen, making them passive participants. According to various research works (e.g., [21], [22]), the root causes of this type of behaviour are culture related. It is promising that the past few years have seen a slow, but steady increase in higher and secondary institutions in Japan encouraging (and often requiring) active participation of students in class and active learning in general.

b) Years 2017-2018

In the subsequent two years, along with the regular practice of asking questions for the purpose of eliciting class discussions, one session each year was used to experiment with the concept of "flipped classroom"). In order to observe the impact on overall class performance, a questionnaire was administered at the end of the class; in it, the students answered questions about their preference for different class styles: lecture style, discussion style, combination (of lecture and discussions) and flipped. Although it proved to be the least popular with the students, the experimental flipped classroom session was perceived as both more challenging and more enjoyable (as shown in [23]).

The instructor introduced several class activities, which appeared to elicit a certain level of enthusiasm, in particular with the international students, who made up the majority of class participants (18 students out of 26 in 2017 and 24 students out of 35 in 2018). In these activities, the students were required to give solutions to a set of given small-scale problems arising during the software development process. They were also involved in replicating certain activities that need to be implemented in the real world, during various phases of software development (again, small scale problems, that can be easily replicated in the classroom).

Moreover, some activities required the creation of mini-teams, which were given tasks to solve during classes. In order to complete their tasks, the students needed to communicate and collaborate with their colleagues/team-mates. After

completing their task, each team chose a representative to report the results in front of the classroom. Our previous works ([24] and [25]) show how collaborative learning was used and the lessons learned during this time.

Considering that the class componence was multicultural, whenever teams were created, the instructor made sure, as much as possible, that the members belonged to different cultural groups. This approach enables the students to leave their comfort zone and helps them acquire a different perspective. In term of the language used, the instructor insisted that the interactions take place in English, thus making sure that everyone understands everything that is being discussed in a team. However, as it turned out, not all team interactions took place in English, even in groups of students with different mother languages. This often happened because the majority of the Chinese students, besides being proficient in English, often speak Japanese fluently, as well. Considering the traditional reluctance of the Japanese students to speak other languages, their Chinese colleagues usually switched to Japanese, in order to facilitate a smoother communication in the team. Nevertheless, it was understood that the reporting in front of the whole class should take place in English. The drawback of this requirement was that the reporting student was seldom a Japanese student; however (as already shown in [26]), the instructor made sure to elicit a minimum level of verbal participation in English from the Japanese students, as well (in the form of brief new comments or at least approval/disapproval of colleagues' comments).

c) Year 2019

A number of 66 students enrolled in the course in 2019. The class activities were continued, with an increased frequency; often, several activities were included in one class. Furthermore, a micro-project was introduced, handled through the learning management system (LMS) in use at the university, i.e., *manaba* [27]. The LMS facilitated the creation of teams, whose members were assigned by the instructor. Items submitted in the project on the LMS could be either visible by the team only or by the whole class. The purpose of the micro-project was teaching the students about the issues that arise during the requirements elicitation phase in the development of a software product and how cultural differences impact the creation of requirements documents. The students' tasks were divided into three parts. In the first part, two members acted as "customers" and 4-6 members acted as "developers". The "developers" created a questionnaire which the "customers" answered in writing. A requirements document was created, using "shall" and "should" items, for necessary and desirable requirements, respectively. In the second part, the "customers" were swapped with those from a different team, after which they gave feedback to the requirements document created by the "developers". In the final part, the requirements documents were shared and discussed with the whole class.

The instructor observed carefully the interactions between students during the implementation of this mini-project. Collaboration between team members played a crucial part and cultural differences had to be overcome at this stage. On one hand, a group of assertive students could be observed, who led the activities and decided the manner in which the mini-project would be conducted. On the other hand, the students who appeared less confident had to overcome their fears and respond/react to the demands of their more assertive colleagues. The team componence was carefully selected again: students from different cultural groups, with different mother languages were assembled in one team.

The benefits of this activity were two-fold: highlighting cultural differences in the classroom (while learning) and depicting cultural differences in the workplace (while developing a software product).

Based on the feedback gathered from the students at the end of the class, along with the additional questionnaires administered by the instructor, this edition of the course was the most successful so far, in terms of implementing active and collaborative learning. As one student stated, *"the lecturer changed the students' silence into discussion and projects"*. Indeed, based on the instructor's observations, the course was successful in persuading the students more inclined to be silent in class to participate more in the various discussions and activities. Notably, several students admitted that what they learned would be useful later, not only in software engineering, but in other fields, as well (to quote one other student, "*the principles that I learned in the class changed the way I approach problem solving in general"*).

To summarize, these first three years of teaching the course featured the use of active and collaborative learning on an increasingly larger scale. Sudden and major changes were about to happen in the following year, as described in the next section.

B. Online Format

As mentioned earlier, the Covid-19 pandemic brought with it an abrupt change in the method of delivering instruction to students and pupils all over the world. According to Whittle at al., the focus was shifted towards "the method of delivering instruction rather than the learning goals" [28], making the implementation of collaborative and active learning even more challenging than usual. The following will describe how these concepts were implemented during the pandemic.

a) Year 2020

The software engineering course starts in April every year; the spring of 2020 represented the beginning of the pandemic and the time when decisions regarding implementation of classes had to be taken. All classes were switched to an online version; there was little time to consider the consequences and there was no realistic view of what the (near) future would bring. In this year, 34 students enrolled in the class (a rather large drop in number from the 66 participants in the previous year). Two of the reasons for this relatively low number could be explained as follows.

First, in the author's opinion, in March 2020 (when students had to register for their spring semester courses), there were many uncertainties regarding the online environment, seconded by an optimistic expectation of return to face-to-face classes in the very near future. These factors enabled the students to think that there was no need to

participate in an "online" class, when a "real", face-to-face one would be back "soon enough".

Second, in our university, students have around two weeks to observe a course, after which they must make a final decision regarding whether they enroll in the course or not. During the first class in 2020 (held online), the students were told that an active participation is required for the duration of the whole course. Moreover, being in an online environment made it more difficult for them to choose their discussion partners. As shown in [25], in case of a physical classroom, the students choose where to sit (usually, next to familiar faces, friends or colleagues who share the same mother tongue). At least for the first two or three lectures, they are allowed to discuss and form groups with the students seated nearby. After a few classes, the instructor specifically asks them to be part of multicultural groups. By this time, the students are familiar with their colleagues; moreover, even if they wanted to, they could not drop the class at this point. In this new virtual environment, several Japanese students felt that combining the difficulties of sudden online learning with the requirement of being active participants in a multicultural class represented a hurdle impossible to overcome. As result, more than 10 students cancelled their registration immediately after the first class [25].

In this first edition of online course, the classes were held synchronously, using Microsoft Teams [29] and Zoom [30]. All class materials were placed on *manaba* (the LMS used in our university [27]). Notably, several international students were still abroad at the beginning of the course (they could not arrive in time or were not allowed to enter Japan, due to the pandemic restrictions). In order to accommodate the time differences and different locations, the classes were also available on demand: each lecture was recorded, and the recordings were placed on Microsoft Stream [31] (with links to recordings placed in *manaba*). At the end of the course, the instructor administered a questionnaire, to find out the students' opinions and perceptions regarding online classes, active learning and specifics of the course they had just taken (the results were summarized in our previous work in [24]).

At the beginning of each class, which started with a warm-up discussion (usually a piece of technology-related news), the students were gently encouraged to turn on their cameras. Some students did that in the beginning, but, as the class progressed, they became more reluctant to speak or show their faces. Moreover, several students became distracted, often engaging in a completely different activity. This was easily observed by the instructor, either when she tried to address them directly and there was no response, or when there were group activities and certain students were not present in the newly created online groups.

The instructor continued to make efforts to involve all students in the learning process. Although conversations online were more difficult to implement than in a classroom, she asked questions and attempted to engage all students in the discussions. Various class activities were adapted to an online format. Instead of teams created in a classroom (organized based on the physical location of the desks), the instructor created breakout rooms in Zoom; each such room acted as a group, where participants held discussions and performed various tasks given in class. (Recording of the meeting was paused during the switch to breakout rooms.)

After organizing the students in breakout room and "opening" the rooms, the instructor took turns visiting each breakout room. She was initially muted, merely observing how the activity is being conducted. If the group appeared to be silent or not active, she would gently guide the activity, suggesting questions, and, if necessary, addressing the non-participative students directly or guiding the group towards deeper discussions. If important issues were raised in a group and the instructor believed that they can be shared with the whole class, or she observed that certain clarifications are needed, she would "call" the participants to the main room and address the issue immediately. (The breakout rooms would be reopened later, as necessary.)

It is worth noting that, in the online version of the classes, more time is needed to organize activities. Switching from the general meeting to breakout rooms and back takes a certain amount of time; thus, planning all the activities in advance is essential.

As previously described in [1], dealing with a multicultural classroom (with 6 Japanese students and 28 from 4 other countries) meant that cultural differences had a strong impact on discussions. Often the same students responded to questions every time in the main meeting. However, within the breakout rooms, it was easier to involve the less communicative (or less confident) students. The most obvious reason is the number of peers present: breakout rooms consisted of only 5-7 students (as opposed to over 30 in the main meeting), which made it easier to overcome the lack of confidence, particularly regarding language skills (English was not the mother tongue for most participants). The majority of students agreed that active participation in an online setting may be more difficult, as can be seen from the following two responses.

- *"I feel like when classes are held online, people will be very hesitant to participate unless they are picked on directly."*
- *"I think the online environment keeps many people silent, or because they don't speak and no one can see so they keep silent. Such discussions are not very effective and there will be problems in the allocation of discussion time."*

Nevertheless, one participant expressed a different opinion, stating that asynchronous work allows for different modalities of work:

- *"I think that online classes can result in *more* discussion than face-to-face classes because students can work asynchronously and in different modalities. I don't need to see someone or hear them to discuss, there are other ways of communication."*

Importantly, when questioned about the class activities, about 90% of the students stated that they found them useful (a lot or in a moderate amount), whereas about 80% found them enjoyable (a lot or in a moderate amount).

Unfortunately, not all previously used activities could be adapted for an online environment, with some of them having to be eliminated. One example is the agile game "paper

airplanes" [32]. This game teaches the benefits of teamwork, working in sprints, planning and retrospectives. It includes the task of physically creating paper airplanes and flying them in the classroom; even though it is considered a "classic" game to illustrate agile development, there was no way of adapting this game to the online environment.

In this first year of online teaching, during the first few classes, one of the most challenging parts proved to be the collaborative part. At first, after providing a task to be completed, a generic request was made by the instructor in each breakout room, for one student to share their document (which could be in almost any format desired by the group), listen to the other group members' opinions, and take notes (draw diagrams, write text, etc.). Often, no student was willing to take the initiative, to guide the discussions or to share their own screen. In later classes, the instructor did not ask for a volunteer, but instead designated a student to be the "sharing" member, and this proved to be a more effective strategy of involving students.

Despite the sudden changes in class format and the initial worry that the classes would be much less interactive than usual, the instructor believes that active and cooperative learning were implemented to an acceptable level, under the given circumstances, i.e., in the ERT format.

b) Year 2021

In 2021, with the pandemic situation not improving, the software engineering course was held online again. A number of 53 students enrolled in the class - an increase of more than 50% from the first online class. Considering that students often consult with their seniors on which classes to enroll in, a larger number of students might hint to the impression that the online edition was rather successful and the students this year were encouraged to take it, despite its new format. Not only more international students enrolled (37, from 9 different countries), but the number of Japanese students increased, as well (from 6 to 16 participants).

The format used was very similar to the one from the previous year. Classes were held online, synchronously, either on Zoom or on Microsoft Teams. They were recorded and made available offline for the students, through the LMS. By now, the instructor had a better idea of what could work in an online setting, and she organized even more class activities. Some of these activities were adapted from those originally used in a face-to-face setting, other were simply new ones.

To facilitate the completion of the group task and to make sure that important ideas exchanged in a group are recorded, an online document was shared, with sections available for each group in the breakout room. This idea came from a student, who noticed that it would be beneficial for the whole class, not only the group involved. In this way, anyone could edit the document, and anyone could see what other groups wrote and what they were working on. Often students who completed their task in their own group would browse the common document and comment on other groups' work. A certain level of competition could be observed; allowing the participants to see other people's work made them try to improve their own work. From this point of view, this was an improvement from the offline class modality: in a classroom, students could not see in real time what other groups are working on, unless their notes were presented in front of the whole classroom, at the request of the instructor.

Just like every year, the instructor gathered feedback from the students with regard to the course. At this point, 40% of them stated that they preferred online classes, whereas almost 25% preferred face-to-face classes. Interestingly, 30% of the students responded that this depends on the class they are taking (see Figure 2).

One participant stated that *"No matter what kind of classroom form, the activity of the classroom is very important."*. Another student responded with *"I think "Class Activity" are interesting. But I am Japanese and not good at English. If I could speak English very well, I would have been able to participate more actively."*. Other comments included *"The class with discussion activity should be Face-to-Face."* and *"Face-to-face classes are more interactive and engaging"*, supporting the idea that discussions and class activities may be perceived as more successful in a physical classroom setting.

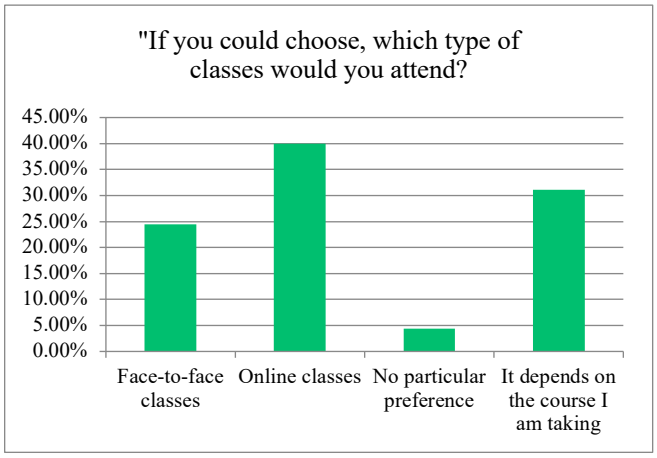


Figure 2. Format preference for attending classes in 2021

The participants were further asked which format they consider more successful for class activities/discussions. About 43% considered that they are more successful if held face-to-face, approx. 27% stated that they are about the same, and just over 13% believed that they are more successful if online (the remaining ~17% responded that they do not know).

When asked to compare online classes with face-to-face classes in terms of cultural differences, almost 49% of the students considered that cultural differences are more visible in face-to-face classrooms and about 15% thought that they are more visible in online classes. 20% of the students found no difference between the two (with the remaining 16% stating that they do not know).

Last, but not least, the students were questioned whether they find the class activities useful/valuable on one hand and enjoyable on the other hand. The results are summarized in Table II. As can be observed, more than three quarters of the

students find these activities enjoyable and more than 90% find them valuable (either a lot or moderately). These results show us that, despite the cultural differences and the difficulties inherent to online environments, the students generally found the active learning implementation not only useful, but also rather enjoyable. The instructor's observations are in line with these results: based on the impressions gathered in class, she felt that the course was successful and that the activities and discussions played an important role in achieving this.

TABLE II. CLASS ACTIVITIES IN 2021: ENJOYABLE VS. VALUABLE/USEFUL

	A lot	*Moderately*	*A little*	*Not at all*
Enjoyable	24.44%	53.33%	20.00%	2.22%
Valuable/useful	32.11%	60.00%	8.89%	0%

c) Year 2022

This past year, the course proceeded very similarly to the one in 2021. One might wonder why there was no return to the physical classroom yet: Japan was still rather strict with its pandemic related rules and there were still students who had not managed to enter Japan. The decision was made to continue with mostly online classes. Coincidentally, the number or students enrolled was the same as in 2019, i.e., 66; similarly, the number of international students was exactly the same as in 2019, i.e., 33 students. By this time, all the students were familiar with the online environment, Microsoft Teams and Zoom (as well as *manaba*, i.e., our university's LMS).

With the official course team created on Microsoft Teams, the first few classes were conducted on this platform. During subsequent discussions with students, it was revealed that several of them have a strong preference for Zoom; one session was therefore conducted on this platform, at the students' request. However, the remaining classes took place on Teams – a decision taken by the instructor together with the students. There were minor differences between how classes took place on the two platforms. One of them is that the students could easily choose their display name on Zoom, but they could not change their already assigned display name on Teams. This name was decided when they first signed up for Microsoft 365, as part of the university's organization. In the case of Japanese and Chinese students, their names were often written in "kanji" (Chinese characters). For the international students who cannot read them (and considering that names are particularly difficult to read), this raised problems in recognizing their colleagues' names in the group.

Whereas the course was held again online, synchronously, one notable difference from the previous year is that the instructor decided not to record the classes. The assumption was that there were no particular factors preventing the students from attending the classes in real time. Moreover, the instructor's (undeclared openly) intention was to make sure that students attend every week and pay attention in class, as much as possible (thus not relying on class recordings).

Most of the class activities were similar to those from the previous year. There were slight differences in the level of participation, i.e., the international students seemed to blend slightly better with the Japanese students. However, this is most probably a very personal aspect – different students with different personalities participate in each year. The groups were created by the instructor and seldom randomly assigned. This year, the students were allowed to use any language they wanted in the breakout rooms, as long as they made sure that everything that was said was understood by all the group members. This meant, for example, that Japanese was used in a group where everyone spoke the language (either Japanese students only or a mixture with other students proficient in Japanese). English remained the main language of the course, as well as the language used for plenary discussions.

At one point, the instructor conducted a poll in which students were asked if they prefer language-based groups, randomly assignment groups or if they simply believe that this aspect does not matter. Less than a third of the respondents preferred language-based groups, with another third preferring randomly assigned teams; the rest of the students simply stated that it did not matter. This was a pleasant surprise for the instructor, who had assumed that the students would insist on being able to speak their own language. However, it is also possible that the meaning behind these preferences is simply that, at least in case of some of the students, they did not care about the groups because they did not want to participate anyway.

After the course ended, the instructor asked the students to answer a questionnaire about their experiences with the online software engineering class, to which 33 students ended up answering (this number represents exactly half of the total number of participating students). At first, the students responded with regard to their preference for an online or face-to-face class; their answers are summarized in Figure 3.

We can observe that the general preference of the students has changed compared to last year. At this point, most students prefer face-to-face classes (*"Face-to-Face would be more productive"*): more than 40%, compared to 25% last year. As for online classes, the percentage of students who prefer them decreased from 40 in 2021 to slightly more than 30. Same as the previous year, several students stated that their preference depends on the course they are taking (almost 20%).

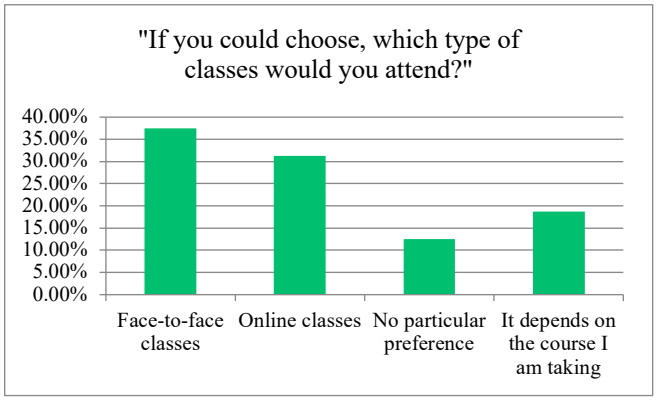


Figure 3. Format preference for attending classes in 2022

Some of the observations that students made reflected their feelings toward different formats. One student commented on the convenience of online classes, but used the term "invisible" to show that this is detrimental to the presence in the group: "*Although online classes are convenient, I think it does not help regarding student participation. One would think that not showing yourself would be a plus, but in fact it is used to be even more invisible".*

One advantage of being in an online environment was highlighted, with regard to the seating in a classroom: *"I can speak up and ask questions in a flat environment, regardless of whether I am in the front or back of the classroom.".* Unfortunately, with all the restrictions and their impossibility to arrive to Japan, there were students who had never been in a face-to-face class at the university, so they could not make a justified comparison. As one student stated, *"I […] never took face2face class since I was here, so it's hard to justify."*

Next, the class participants were asked to express their preference for the class style used in the software engineering course (see results in Figure 4). They had the following choices: "Lecture" style: teacher speaking in front of the whole class, students listening" (27.27% respondents), "Discussion" style: new things are taught/learned through continuous discussions between students and teacher (21.21% respondents), "Combination": half lecture-style teaching by the instructor, half interactive communication/discussion with students (48.48%) and "Flipped classroom" style: students read the new materials at home, then they come to class and ask questions, discuss etc." (3.03% respondents). Almost half of the students chose the "combination" style, which is precisely the style employed by the instructor for this course. This is a promising result, showing that the class was relatively successful, and the students were satisfied with the style used.

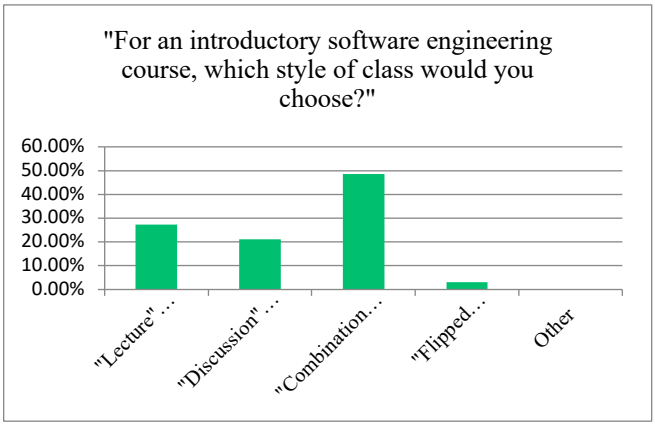


Figure 4. Preference for class style

When questioned about cultural differences in face-to-face classes vs. online classes, the responding students considered the two formats equally influential in how visible these differences are (as results show in Figure 5). Almost 40% of the students considered that cultural differences are more visible in face-to-face classrooms; the same percentage stated that they are more visible in online classrooms.

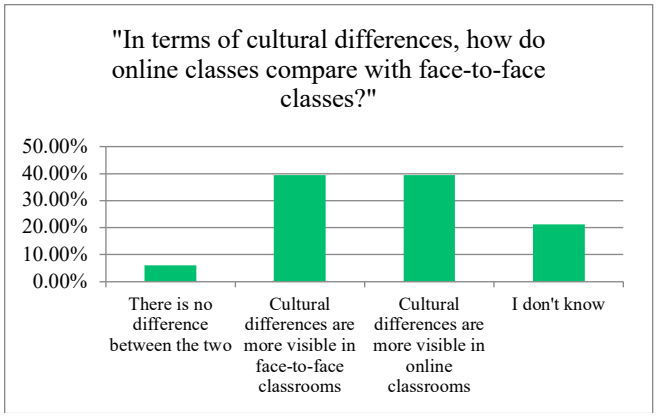


Figure 5. Differences between online classes and face-to-face classes in terms of cultural diferences

Only 3% of the students found that there is no difference between the two formats. This is an interesting result which shows how the perceptions of online classes have changed for the participating students. One conclusion that could be drawn is that, at first, students believed that the cultural differences easily distinguishable in a classroom would attenuate in an online environment. After two years of almost exclusive online classes, their perceptions seem to have changed: they realized that the differences not only remained when online, but they are often even more difficult to overcome.

The class activities are an important part of implementing active and collaborative learning. In fact, from the point of view of the instructor, this is one of the greatest improvements that she managed throughout the years of teaching the course. Every year, more numerous and more diverse activities were introduced. The instructor wanted to know whether the participating students enjoyed these activities on one hand, and whether they found them useful/valuable on the other hand. Furthermore, she wanted to learn precisely what the students liked and what they did not like about these activities. Table III summarizes some of the data obtained from the questionnaire administered to the students.

TABLE III. CLASS ACTIVITIES IN 2022: ENJOYABLE VS. VALUABLE/USEFUL

	A lot	*Moderately*	*A little*	*Not at all*
Enjoyable	45.45%	33.33%	21.21%	0%
Valuable/useful	60.61%	27.27%	12.12%	0%

As can be observed, more than three quarters of the responding students found the activities enjoyable (either a lot or moderately). At the same time, almost 90% thought that they were valuable/useful (a lot or moderately). The results are similar to those from 2021 and, again, they show that students do recognize the usefulness of the activities even when they do not enjoy them very much.

The respondents were given 3 choices in regard to what they like about activities. Most respondents (63.64%) chose "listening to what other people think", followed by "saying what you think", which was chosen by 24.24% of the respondents. Some students (12.12%) chose "nothing" (simply not enjoying the discussions at all). When asked to specify what they did not like, several responses stood out. By far, the largest complaint expressed regarded the participation of the other students (often Japanese), as seen below.

- *"I felt like I had to force discussion with others sometimes."*
- *"Limited number of people speaking up, only two or three may participate in team activities"*
- *"That Japanese people do not interact much in the class."*
- *"After we're divided into breakout rooms, some people don't join the discussion. I don't know why, but I want them to join."*
- *"I wish all the students participate in the discussion so that I can interact with them.:*
- *"Many students do not speak."*
- *"Just some people not being involved in the class 100% of the time."*

With the same complaint, some students' opinions included more details:

- *"I like the discussions, but sometimes I'd prefer to not have this kind of activity because of the number of times everybody stays quiet in my room. I feel I waste more energy trying to do the exercise alone or getting some else to talk. I understand everybody has its reasons, but these situations are hard for me."*
- *"It is hard when one's put in a room where nobody wishes to talk. (Now I kind understand how the professor feels) Although everybody may have their reason, it affects how I enjoy the class as I think I waste more energy: Doing the exercise alone or trying to get someone else to talk. I'd say it even gave me a little be on anxiety sometimes, as sometimes I noticed that if I didn't start to make the requested exercise, nobody in the group say or do anything."*

One interesting piece of feedback highlighted the fact that some students "talk too much on irrelevant topics: *"i. people won't talk in breakout rooms ii. some would talk too much on irrelevant topics".* From the instructor's point of view, in a class where most Japanese students are very quiet, talking "too much" is beneficial for the class, even considering that the topics are "irrelevant".

One respondent's comment referred to one of the sessions when the groups were constituted on the basis of a common language (as much as possible): *"Maybe we didn't do ice breaking properly, and when we were in breakout room, no one speak. There were some classes that there was a real discussion. The first one is when we group student based on language we speak, and the last one that we discussed about risk assessment. About the first one, I think it because of there were so many Latin American students which shared a lot in their culture so it was easy to connect for them. I think if everyone know each other more, we are likely to share a conversation."*

Not all the students were eager to take part in class activities, even though they appreciated the content, as one participant explained: *"The content of the class was very useful. Only in my point of view, I think that maybe not having so many activities together would have helped me."*

Regarding class participation and ways to increase it, one interesting suggestion came from a student: *"Attributing the participation score may motivate more people to participate in the class.".* Whereas this is an issue worth considering, it would most probably be hard to implement. Students joining a software engineering class would expect to be evaluated more on the knowledge they acquired and less on their participation skills.

Another participant pointed out that easier (and more "enjoyable") tasks might entice the students to speak more during class: *"I am shy and not good at speaking and listening English. So, I want to do easy and enjoyable tasks or games, for example, estimating rule from dice results on 6/8. Thanks to this game, I understood and talked in English."*.

A third suggestion came in the form of suggesting "ice breaking" activities: *"Some people (e.g., me) are unlikely to discuss with someone I am not familiar, but will talk a lot with close friends. I think a lot of people are like me. So, I think we need some kind of ice breaking activities. Also, I have not met anyone in my degree since I have been here too.".* (The last part refers to the fact that the student is the only one joining the course from his department ("degree").)

Some participants believe that online classes are helpful for "shy" students: *"online class is more flexible for me, and relax for shy people, i am sure i would only speak less in a face to face class."* At the same time, other participants believe the opposite: *"Offline courses will make shy people more likely to participate in discussions".* In the instructor's experience, it is difficult to generalize: some students are more comfortable participating online, whereas others, although initially "forced" to participate in a real classroom, become increasingly more confident in the presence of other students. Understanding the factors that contribute to better participation and making a systematic comparison between online and offline classes (from the active and cooperative learning point of view) would make an excellent topic that could be studied in future work.

Last, but not least, even in informal discussions, many students expressed their hope that the face-to-face classes will return soon. Here is an example of one such comment: *"I wish the classes to be face-to-face so that I could get a chance to interact with more people and exchange the ideas I have and get their opinions as well. I wish to come out of the "online" class mode and interact with other students directly. Hope this changes soon."*.

Finally, when students had the opportunity to express free thoughts, they made comments like the following:

- *"I am grateful that [she] always listened to the opinions of the students and responded to everything on-time, even though it was online"* (translated from Japanese).
- "*The course hours are interactive and this keeps a student engaging in the classwork".*
- *"It was nice to have an English discussion*" (translated from Japanese).

The only request for change took the following form: "*It would be better if the lecture could be posted to manaba before class*", followed immediately by *"Anyway, thank you for your teaching"*. All the remaining comments received were of a positive nature, expressing students' satisfaction with the way the course was conducted.

V. Discussion on course effectiveness and lessons learned

The changes from offline to online classes provided an invaluable experience, to both the students and the instructor. While it is generally rather difficult to measure learning and its effectiveness very precisely, two of the most common ways to assess learning effectiveness are course completion rates and learner perception. This section discusses these two methods and offers some lessons learned by the instructor during the teaching of this course.

The software engineering course completion rates are shown in Table IV. Although slightly lower than those for the previous 3 three years (but higher than that for the very first year, i.e. 2016), the completion rates for the three "online" years were still very high, with the lowest being 94.11%.

TABLE IV. Completion rates of software engineering course

	2016	*2017*	*2018*	*2019*	*2020*	*2021*	*2022*
Number of students enrolled in course	17	26	35	66	34	53	66
Number of students completing course	15	26	34	65	32	52	63
Completion rate (%)	88.23	100	97.14	98.48	94.11	98.11	95.45

Regarding the students' perceptions, the results obtained from the questionnaire administered by the instructor, as described in the previous section, show that they were overwhelmingly positive towards the class's effectiveness. Furthermore, data from the university-prepared end of course questionnaire results support the same idea. (The University of Tsukuba prepares a short questionnaire at the end of each course, which the students are strongly encouraged to answer, usually during the last class of the course. This questionnaire is independent of the one administered by the instructor, as described in the previous section.)

The students provided their answers on a scale of 1 to 5, corresponding to "strongly agree", "agree", "neutral", "disagree" and "strongly disagree", respectively. Table V summarizes the answers given by the students to 5 of the questions in 2022, the last year of the online classes. As can be observed, with only a few exceptions of respondents choosing "neutral", all the participants selected either "strongly agree" or "agree" with the 5 statements. It is encouraging to observe that participants developed a stronger interest in the subject after taking this course. Also, even though the class took place online, they believed that the ways in which the instructor explained and planned the class contents were suitable for the course. They also recognized that there were sufficient opportunities to ask questions during class. Last, but not least, the students were satisfied with the course, as shown in their responses: 26 chose "strongly agree", 14 chose "agree" and 1 chose "neutral".

TABLE V. Results of end of course questionnaire (1: strongly agree; 2: agree; 3: neutral; 4: disagree; 5: strongly disagree)

	1	*2*	*3*	*4*	*5*
"The instructions were well prepared for the course."	29	12	0	0	0
"The ways the instructor explained and planned the class contents were suitable for the course."	23	17	1	0	0
"Attending this course, I developed a stronger interest in the field of study related to this subject than before."	21	17	3	0	0
"Overall, I am satisfied with this course."	26	14	1	0	0
"You were given sufficient opportunities for asking questions to the instructor(s)."	32	8	1	0	0

Furthermore, empirical observations supported these results. The instructor held informal discussions with the participants, as often as possible (either during the break time or when she met the students on campus, by chance). The students expressed their satisfaction with the course, as well as their enjoyment in participating in class activities, on numerous occasions. They often expressed these feelings even without being asked.

To summarize, some important lessons from our experience as described in this paper are as follows.

a) Active learning should be pursued in all its forms, as much as possible. In software engineering, it paves the way for an easy to achieve understanding of various concepts and allows the learners to gain experience similar to that obtained in the real world when developing a software application.

b) Collaborative learning can be very successful, in both offline and online settings. As long as an environment that facilitates collaboration between participants is ensured, collaborative learning can be implemented in either class modality, be it online or offline.

c) Perceived enjoyment and perceived usefulness play an important role in an online environment for the students. The more enjoyable and useful they feel that classes are, the more engaged and satisfied with the course they will be.

d) Multicultural classrooms provide the opportunity for students to experience work in international environments and a glimpse of global software engineering practices.

e) Fulfilling course objectives and achieving student satisfaction depend less on the class modality, and more on the student engagement strategies used by the teacher.

It is a fact that the traditional face-to-face classroom modality is preferred by many course participants (and even the instructor herself). In the case of the software engineering course, this preference does not rely on the lack of success of online classes, but rather, we suspect, on factors that have to do with human psychology. (This is a topic that can be further developed in future work.) Nevertheless, the most valuable lesson is the fact that, if necessary, online classes can

successfully substitute face-to-face classes, provided that there is constant interaction between the teacher and the students, as well as collaborative activities between the students.

VI. Conclusion and Future Work

Based on the experience of teaching a graduate school software engineering course to a multicultural group of students, this paper described the approaches used by the class instructor to implement active and cooperative learning, in both the traditional face-to-face format and the sudden (imposed by the Covid-19 pandemic) online format. It highlighted the impact of switching from an offline to an online class modality and the challenges brought upon by emergency remote teaching. Our study showed that, despite all the challenges, the online classes were successful in achieving the learning outcomes, as well as achieving student satisfaction with the learning process. If the online class interactions (between the teacher and the students, as well as among students) are constant and the activities are varied, the students participate actively, even after initial reluctance. Whether the class is enjoyed by the students, and they are satisfied or not, depends more on this level of interaction rather than the class modality.

In future work, the instructor plans to make a systematic comparison between the online and offline environments from the point of view of student in-class participation, in order to discover the best strategies for making the best out of active and cooperative learning, in either type of environment.

References

[1] S. Vasilache, “Offline and Online Active Learning: Lessons in Teaching Software Engineering to Multicultural Groups”, ICSEA 2022.

[2] D. Dahiya, "Teaching software engineering: A practical approach", ACM SIGSOFT Software Engineering Notes, 35(2), pp. 1-5, 2010.

[3] B. R. Maxim, S. Acharya, S. Brunvand, and M. Kessentini, “WIP: Introducing active learning in a software engineering course”, In 2017 ASEE Annual Conference & Exposition, June 2017.

[4] M. A. Alsubaie, "Examples of current issues in the multicultural classroom", Journal of Education and Practice, 6(10), pp. 86-89, 2015.

[5] M. Malcon-Cervera and C. Montaudon-Tomas, "Multicultural Classrooms: Advantages for Foreign and Local Students. A Comparative Study", In EDULEARN17 Proceedings, IATED, pp. 6477-6485, 2017.

[6] M. W. Keyser, "Active learning and cooperative learning: understanding the difference and using both styles effectively", Research strategies, 17(1), pp. 35-44, 2000.

[7] S. Sandrone, G. Scott, W. J. Anderson, and K. Musunuru, “Active learning-based STEM education for in-person and online learning”, Cell, vol. 184, no. 6, pp. 1409-1414, 2021.

[8] G. S. Gremmels, “Active and cooperative learning in the one-shot BI session”, Library Orientation Series, pp. 101-106, 1996.

[9] C. C. Bonwell and J. A. Eison, “Active Learning: Creating Excitement in the Classroom”, 1991 ASHE-ERIC Higher Education Reports. ERIC Clearinghouse on Higher Education, The George Washington University, 1991.

[10] S. Freeman, S.L. Eddy, M. McDonough, M.K. Smith, N. Oko roafor, H. Jordt, et al., “Active learning increases student performance in science, engineering, and mathematics”, Proceedings of the National Academy of Sciences, 111 (23), pp. 8410-8415, 2014.

[11] V. M. F. Fonseca and J. Gomez, “Applying active methodologies for teaching software engineering in computer engineering”, IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 12, no. 4, pp. 182-190, 2017.

[12] D. W. Johnson, R.T. Johnson, and K. A. Smith, “Nuts and Bolts of Cooperative Learning”, pp. 1-3. Edina, MN: Interaction Book, 1991.

[13] W. Ali, “Online and remote learning in higher education institutes: A necessity in light of COVID-19 pandemic”, Higher education studies, 10(3), pp. 16-25, 2020.

[14] C. Hodges, S. Moore, B. Lockee, T. Trust, and A. Bond, “The difference between emergency remote teaching and online learning”, Educause review, 27, pp. 1-12, 2020.

[15] W. H. Stewart, Y. Baek, and P. R. Lowenthal, “From Emergency Remote Teaching (ERT) to Sustained Remote Teaching (SRT): A comparative semester analysis of exchange students’ experiences and perceptions of learning online during COVID-19”, Online Learning, 2022.

[16] T. E. Shim and S. Y. Lee, "College students’ experience of emergency remote teaching due to COVID-19", Children and youth services review, 119, p. 105578, 2020.

[17] A. Bozkurt and R. C. Sharma, "Emergency remote teaching in a time of global crisis due to CoronaVirus pandemic", Asian journal of distance education, 15(1), pp. i-vi, 2020.

[18] M. Chierichetti and P. Backer, “Exploring faculty perspectives during emergency remote teaching in engineering at a large public university”, Education Sciences, 11(8), p. 419, 2021.

[19] K. A. Douglas, A. C. Johnston, J. P. Martin, T. Short, and R. A. Soto-Pérez, “How engineering instructors supported students during emergency remote instruction: A case comparison”, Computer Applications in Engineering Education, 30(3), pp. 934-955, 2022.

[20] Y. Y. Chiu, “Effectiveness of online learning relies on the dynamic between teachers and students”, Journal of e-learning Research, 1(4), pp. 61-81, 2021.

[21] H. Mercier, M. Deguchi, J. B. Van der Henst, and H. Yama, “The benefits of argumentation are cross-culturally robust: The case of Japan”, Thinking & Reasoning, vol. 22, no. 1, pp. 1-15, 2016.

[22] R. G. Tweed and D. R. Lehman, “Learning considered within a cultural context: Confucian and Socratic approaches”, American Psychologist 57, No. 2, pp. 89-99, 2002.

[23] S. Vasilache, “From an International Classroom to a Distributed Work Environment: Student Perspectives on Global Software Engineering”, Proceedings of International Conference on Teaching and Learning for Engineering (TALE 2018), Dec. 2018, pp. 825-828.

[24] S. Vasilache, “Suddenly Online: Active Learning Implementation Strategies During Remote Teaching of a Software Engineering Course”, In: Auer M.E., Hortsch H., Michler O., Köhler T. (eds) Mobility for Smart Cities and Regional Development - Challenges for Higher Education, ICL 2021, Lecture Notes in Networks and Systems, vol 389. Springer, Cham, pp. 395-402, 2021.

[25] S. Vasilache, “A Taste of Distributed Work Environments: Emergency Remote Teaching and Global Software Engineering”, In: Stephanidis C., Antona M., Ntoa S. (eds) HCI International 2021 - Posters. HCII 2021, Communications in Computer and Information Science, vol 1421, Springer, Cham, pp. 624-628, 2021.

[26] S. Vasilache, "Ad Meliora": Towards an Improved Approach to Global Software Engineering Curriculum. In HCI International 2020-Posters: 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22, Springer International Publishing, pp. 343-348, 2020.

[27] Manaba [Online]. Available from: https://manaba.jp/products/ 2023.06.01

[28] C. Whittle, S. Tiwari, S. Yan, and J. Williams, "Emergency remote teaching environment: a conceptual framework for responsive online teaching in crises", Information and Learning Sciences, 2020.

[29] Microsoft Teams [Online]. Available from: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software 2023.06.01

[30] Zoom: Zoom Meetings & Chat [Online]. Available from: https://zoom.us/meetings 2023.06.01

[31] Microsoft Stream [Online]. Available from: https://www.microsoft.com/en-us/microsoft-365/microsoft-stream 2023.06.01

[32] B. Willmott, "The Agile Paper Airplane Game". [Online]. Available from: https://www.ppm.academy/post/the-agile-paper-airplane-game 2023.06.01

An Optimal PID Based Trading Strategy under the log-Normal Stock Market Characterization

Vadim Azhmyakov
College of Computing
Prince of Songkla University
Phuket, Thailand
email: vazhmyakova@phuket.psu.ac.th

Ilya Shirokov
Algorithmic Systems Corp
Phuket, Thailand
email: iyushirokov@gmail.com

Yuri Dernov
Algorithmic Systems Corp
Phuket, Thailand
email: dernovyua82@gmail.com

Luz Adriana Guzman Trujillo
LARIS, Université d'Angers
Angers, France
email: luz.guzmantrujillo@etud.univ-angers.fr

***Abstract*—Our paper proposes an optimal trading algorithm based on a novel application of the conventional Control Engineering (CE) to Algorithmic Trading (AT). We consider a fundamental CE concept, namely, the feedback control and apply it to the algorithmic trading (algo trading). The concrete feedback control strategy is designed here in a form of the celebrated Proportional-Integral-Derivative (PID) model. The highly frequent nature of the modern financial markets motivates the using of a model-free realisation of the generic PID framework. The control theoretical methodology we propose is additionally combined with some advanced statistical results for the historical market data. We obtain a specific log-normal probability distribution function (pdf) associated with the specific relative characteristics of the available stock data. This empirical pdf mentioned above provides a novel computational technique for the necessary PID gains optimization. For this aim we apply a suitable data driven optimization problem and also consider an alternative stochastic programming framework. The stochastic optimization naturally involves the Monte Carlo solution procedure. The optimized PID trading algorithm we propose is next represented in the frequency domain. This equivalent representation makes it possible to introduce a new concept in the financial engineering, namely, the "stock market energy" concept. Finally, we implement the resulting PID optimal trading algorithm and develop a Python based prototype software. We apply the corresponding prototype software to the Binance stock market. This practical example illustrates the proposed optimal PID trading scheme and also shows the effectiveness of the CE methods in the modern AT.**

***Index Terms*—financial engineering; feedback algorithmic trading; model-free PID control; PID gains tuning; data driven optimization; forward testing; stochastic optimization; prototype software**

I. Introduction and Motivation

Optimal design of profitable trading algorithms for financial markets constitutes a very challenging and technically sophisticated problem of the modern financial engineering (see e.g., [1],[3],[4],[5],[7],[8],[9],[10],[11],[31],[38],[41]). In this contribution, we study an idealized stock market and apply the CE based PID control design in the development of an optimization based trading algorithm. Recall that the PID synthesis constitutes one of the most powerful and successful real-world control algorithms (see e.g., [2],[27],[28],[34],[36]). The conventional model-based PID control implements a fundamental idea of the feedback action and is widely used in various engineering and applications.

The stock market idealization we follow in this paper usually involves some generic technical assumptions. We consider the situation characterized by "no transaction costs" and also assume the condition of "one stock portfolio". We also consider some further simplifying hypothesises, namely, the assumption of the "zero interest" and the "continuous trading" condition. Let us refer to [7],[8],[9] for the necessary technical details on the existing abstractions in the modelling of stock markets. Our paper deals with the discrete-time dynamics of the financial markets under consideration. This assumption is motivated by the generic stock tick dynamics and also by the corresponding decision making process. Recall that a tick is a measurement of the minimum upward or downward movement in the price of a security (see e.g., [24],[32] and the references therein). The given discrete dynamics of the stock market ticks naturally implies the discrete-time decision making in trading. We next use the generic notation of the stock ticks $t = 1, ...,$ and the corresponding semi-open time-intervals of the trading buckets $[t, t+1)$.

The advanced financial engineering and financial economics constitute nowadays a powerful theoretical tool of the modern financial science (see [15],[16],[23],[40]). On the other hand, the mathematically rigorous time-series based financial theories can not be directly applied to an algorithmic generation of a profitable trading decision. This fact is a simple consequence of the highly frequent stock price dynamics that makes it impossible any adequate price forecasting. This fundamental property of the modern financial markets implies the so called High-Frequency Trading (HFT) approach. We refer to [16] for the corresponding concepts and useful information. A successful HFT strategy development is a very sophisticated challenging task. In this paper, we propose to use some fundamental aspects of the conventional CE design for this

purpose. More specifically, we propose to use an optimized PID type control strategy for a profitable trading algorithm. The optimization techniques for this PID trading algorithm involve some novel optimal PID gains tuning strategies.

Recall that the control theoretic approach to the modern AT was originated in [7],[8],[9],[10],[11],[19],[31]. Let us also refer to [3],[4] for some novel PID related trading algorithms with a switched structure. Since the non-regular, highly frequent bid-ask spread behaviour on a stock does not admit a realistic forecasting model, we examine here a specific model-free version of the classic PID control. Our algorithmic approach proposes to react to the stock price variations instead of modeling them. We next combine the conventional PID control approach to AT with an advanced statistical characterization of some historical market data sets. We next identify a log-normal type probability distribution function (pdf) for some concrete quantities related to the data sets under consideration.

It is common knowledge that a classic and advanced PID control design incorporates a very important technical step, namely, the PID gains tuning [2],[27],[28],[34]. The established log-normal pdf for some market characteristics can next be used in the optimal PID gains tuning procedure. Note that various optimization techniques play an important role in the modern financial engineering (see e.g., [1],[4],[5],[14],[22],[26],[29],[30],[33],[42],[43],[44] and references therein). For the concrete PID gains optimization we use here two conceptually different approaches. The first approach constitutes a data driven regression based optimization. This useful approach can be combined with the well-known Forward Testing methodology and with the consideration of the corresponding In-Sample and Out-of-Sample concepts for the data sets. The second optimization approach we use is the classic stochastic optimization framework. This approach uses the established log-normal distribution of some market related quantities. It finally leads to the celebrated Monte Carlo solution technique (see e.g., [22],[39] and references therein).

The resulting PID trading algorithm with the optimized gains tuning procedure was applied to some concrete real-world examples. We illustrate the efficiency, profitability and practical implementability of the proposed algorithm and consider a trading application on the Binance BTC/USDT spot market. We also discuss shortly the necessary prototype software for implementation of the proposed optimal PID trading algorithm.

The remainder of our paper is organized as follows: Section II contains a formal AT problem statement. This section includes a mathematical description of the conceptually novel model-free PID trading algorithm. In Section III we perform an advanced statistical analysis of the stock market data. We establish some log-normal probability distribution properties for some specific subsequential market characteristics. Section IV deals with the main problem of the PID trading, namely, with the optimal PID gains tuning. We consider the data driven optimization involving the historical data and the corresponding backtesting procedure. The corresponding regression analysis is considered in the general Forward Testing framework. In this section we also propose an alternative PID gains selection and use the scenario based stochastic optimization problem for this purpose. The resulting stochastic program is next solved by the classic Monte Carlo approach. In Section V we discuss a novel frequency domain interpretation of the proposed PID trading strategy. We apply the conventional z-transformation and the Fourier transformation for the constructive characterization of the PID trading scheme in the frequency space. Section VI contains a practically motivated application of the developed PID trading algorithm to a specific stock market, namely, to the Binance BTC/USDT futures stock. This section is also devoted to the prototype software design. Section VII summarizes our paper.

II. Model-Free PID Based Trading Algorithm

Consider a trading on an idealized stock market with an initial deposit (initial investment) $I(1) = I_1$ and introduce the current return

$$\Delta g(t),\ t = 1, ..., .$$

The current investment level at a time instant t is denoted by $\Delta I(t)$. In parallel with the current values introduced above we also consider the cumulative return and the cumulative investment $g(t)$ and $I(t)$, respectively. The nonlinear discrete-time PID trading strategy can be formalized as follows:

$$\begin{aligned} &\delta I(t+1) = K_P(t+1)\Delta g(t) + K_D(t+1)\dot{\Delta} g(t) + \\ &K_I(t+1)\int_{t-T}^{t} h(\tau)\Delta g(\tau)d\tau, \qquad (1) \\ &\Delta I(t+1) = \chi(\delta I(t+1)), \quad \text{for} \quad t = 1, ... \end{aligned}$$

By $K_P(\cdot)$, $K_D(\cdot)$ and $K_I(\cdot)$ we denote here the necessary PID gains for the proportional, integral, and derivative terms of the classic regulator scheme (see [2],[27],[28]). Let us introduce the vector of these PID gains:

$$K(\cdot) := (K_P(\cdot),\ K_D(\cdot),\ K_I(\cdot))^T.$$

Note that the integral term in (1) incorporates the "process memory" on a given time interval $[t-T, t]$. The integral kernel $h(\cdot)$ in (1) is defined by a suitable "memory loss" function. We consider an exponentially weighted "memory loss" function $h(\cdot)$ with $h(t) = 1$. Note that the proposed scheme (1) is similar to the classic PID control design (see e.g., [28],[34]). The nonlinear function $\chi(\cdot)$ in 1 constitutes a "control saturation" and can naturally be defined as follows:

$$\chi(\delta I) := \begin{cases} \delta I, & \text{if } \delta I^{\min} \le |\delta I| \le \delta I^{\max}; \\ \pm\delta I^{\max}, & \text{if } |\delta I| > \delta I^{\max}; \\ 0, & \text{if } |\delta I| < \delta I^{\min}. \end{cases} \qquad (2)$$

Here

$$\delta I^{\max},\ \delta I^{\min}$$

are prescribed maximal and minimal current investment levels, respectively. The above "saturated" investment model constitutes a formal mathematical condition and serves as a natural restriction for the investment volume.

Assuming the investment decision $\Delta I(t+1)$ in (2) and taking into consideration the stock price $p(\omega,t+1)$, we next calculate the current return $g(t+1)$:

$$\Delta g(t+1) = \frac{(p(\cdot,t+1) - p(\cdot,t))}{p(\cdot,t)} \Delta I(t+1) \tag{3}$$

If we use the CE analogy, we conclude that the proposed investment level $\Delta I(t+1)$ in (1) plays a role of a "control input". We next call it "investment decision". Note that the current investment decision is deployed at a present time instant t under the condition of a natural unknownness of the market price

$$p(\omega,t+1).$$

By $\omega \in \Omega$ and Ω we denote here a probability state space with a unknown probability measure. This stock price

$$p : \Omega \times \mathbb{Z}_+ \to \mathbb{R}$$

is assumed to be a measurable stochastic function. Note that the current return $\Delta g(t+1)$ is calculated an a posteriori value, where $p(\cdot,t+1)$ in (3) denotes a concrete realization of the stochastic price $p(\omega,t+1)$.

Since we have a CE analogy, we also can represent the above PID trading algorithm (1)-(3) using the block diagram. The corresponding block scheme of the proposed model-free PID trading strategy (1)-(3) is presented in Figure 1.

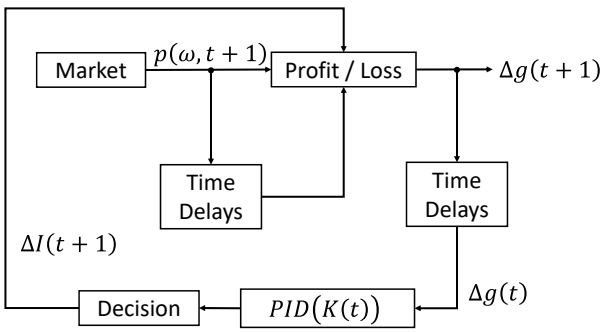


Fig. 1. Model-free PID based trading algorithm

Note that the feedback channel in (1)-(3) is implemented by the current return $\Delta g(t+1)$ in (3). In fact, the PID trading algorithm (1)-(3) and the corresponding block diagram (Figure 1) represent the so-called "delayed PID" scheme. We refer to [28] for details.

The general integral formula for

$$\delta I(t+1)$$

in (1) can be concretized in the discrete-time:

$$\begin{aligned} &\delta I(t+1) = (K_P(t+1) + K_I(t+1) + K_D(t+1))\Delta g(t) + \\ &(K_I(t+1)h(t-1) - K_D(t+1))\Delta g(t-1) + \\ &K_I(t+1)\sum_{\tau=t-T}^{t-2} h(\tau)\Delta g(\tau). \end{aligned} \tag{4}$$

Recall that the main CE problem of the conventional PID control approach consists in defining the adequate PID gains tuning rules (see e.g., [27],[28] and references therein). In the generic real-world application fields of the classic PID controllers the tuning techniques are well established [2],[34]. For the conceptually sophisticated model-free PID algorithm under consideration, an adequate design of the PID gains tuning schemes constitutes a challenging theoretic problem. The suitable PID gains selection immediately determines the main trading signal by the rule (4). In fact, an adequate (optimal) PID gains tuning scheme development is a key problem of a profitable PID based investment decision.

Let us also note that the model-free character of the proposed PID scheme (1)-(3) and (4) indicates the application possibility of the modern ML approaches to an optimal PID gains tuning problem. We refer to [12],[13],[26] for the technical details related to the RL approach and to some applications of the ML techniques in AT.

III. Advanced Statistical Description of the Stock Market Data

We now study the important PID gains tuning problem and apply an additional statistical information related to the stock market. Introduce the following "price/volume" ratio:

$$\theta(\omega,t+1) := \frac{p(\omega,t+1)}{v(t+1)}. \tag{5}$$

Here $p(\omega,t+1)$ is an unknown (stochastic) price at the time instant $t+1$ and $v(t+1)$ is an investment volume to the same time:

$$v(t+1) := \frac{\Delta I(t+1)}{p(\cdot,t)}. \tag{6}$$

Note that (6) expresses the entities number of a traded financial instrument (for example, futures trading). The comprehensive (a posteriori) statistical analysis of a wide spectrum of stock instruments demonstrates that the pdf of the above value

$$\theta(\omega,t)$$

constitutes a specific log-normal distribution (see e.g., [18],[20],[35],[44] and references therein). Clearly, the real investment volume of a hedge fund, a private trader or of a bank is usually a restricted. In the case of the PID trading algorithm (1)-(3), the maximal investment volume is a direct consequence of the boundedness of $\Delta I(t+1)$ in (1) and (6).

The log-normal pdf related to the financial markets has mainly been studied for option prices. In this connection we also refer to the celebrated Black and Scholes model [15]. A useful discussion on this subject can also be found in [40]. Let us also mention the log-normal pdf's of some volatility involved quotients and other stock market parameters and indices (see e.g., [1],[5],[18],[23]). We next analyze the stationary statistical distribution $\rho_1(\cdot)$ of the quantity $\theta(\omega,t+1)$ introduced in (5):

$$\begin{aligned} &\rho_1(\theta) = \frac{a}{\sqrt{2\pi}\sigma(\theta - s)} \times \\ &\exp -(0.5\sigma^2) \times (\ln(\theta - s) - \mu)^2. \end{aligned} \tag{7}$$

Here $\mu \in \mathbb{R}$ denotes a statistical mean, $\sigma \in \mathbb{R}_+$ is a dispersion and $s \in \mathbb{R}$ denotes a shifting. Note that (7) is often called "three-parameters"

$$\{\mu, \sigma, s\}$$

log-normal distribution (see e.g., [1],[18]). The concrete parameter values of the log-normal pdf $\rho(\theta)$ can be calculated using the backtesting on the historical stock market data. As mentioned above, (7) constitutes an adequate model distribution for the introduced price/volume ratio $\theta(\omega, t+1)$. The quality of this statistic model can be established by application of the Chi-Quadrat test for distributions qualification. We refer to [35] for necessary technical details. For example, one can consider the normalized value

$$\chi^2/q$$

for the above test. Here q is the number of degrees of freedom. The concrete stock market histogram for a monthly Binance BTC/USDT price/volume ratio $\theta(\omega, t+1)$ is depicted in Figure 2.

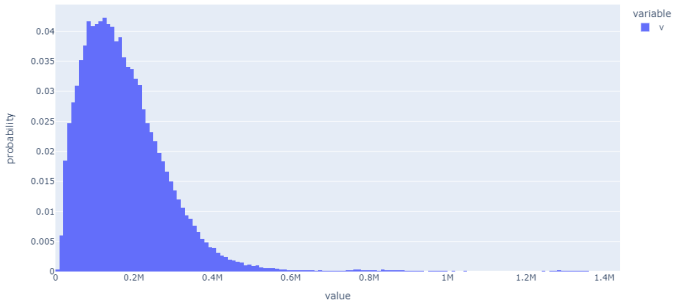


Fig. 2. The log-normal like histogram of the ratio $\theta(\cdot, t)$ for Binance BTC / USDT spot market

In parallel to (7) we also consider the log-normal pdf for the following price/price ratio:

$$\vartheta(\omega, t+1) := \left(\frac{p(\omega, t+1)}{p(\cdot, t)}\right), \tag{8}$$

where

$$\{p(1), ..., p(T)\}, \ \{v(1), ..., v(T)\}.$$

are the stock prices and investment volumes data for the time instants $t = 1, ..., T$. The log-normal probability distribution for the value

$$\vartheta(\omega, t+1)$$

in (8) was established in [1]. We denote this pdf by $\rho_2(\cdot)$. We also refer to [5],[16] for the necessary statistical and econometrical details.

We next obtain

$$\ln\left(\theta(\omega, t+1)/\theta(\cdot, t)\right) = \ln\left(\frac{p(\omega, t+1)}{p(\cdot, t)}\right) - \ln\left(\frac{v(t+1)}{v(t)}\right). \tag{9}$$

and

$$\ln\left(\frac{v(t+1)}{v(t)}\right) = \ln\left(\frac{p(\omega, t+1)}{p(\cdot, t)}\right) + \ln\theta(\cdot, t) - \ln\theta(\omega, t+1). \tag{10}$$

Expressions (9)-(10) make it possible to generate a part of the complete investment decision, namely, the decision about the investment volume $v(t+1)$ to the time instant $t+1$:

$$v(t+1) = \exp\left[\ln\left(\frac{p(\omega, t+1)}{p(\cdot, t)}\right) + \ln\theta(\cdot, t) - \ln\theta(\omega, t+1) + \ln v(t)\right]. \tag{11}$$

Observe that $\theta(\cdot, t)$ and $v(t)$ in (11) are known values. Additionally, the price/volume ration $\theta(\omega, t+1)$ and the price/price ratio

$$\frac{p(\omega, t+1)}{p(\cdot, t)}$$

can effectively be simulated using the corresponding log-normal probability distribution functions. We next can also forecast the value

$$\ln\left(p(\omega, t+1)/p(\cdot, t)\right)$$

in (11) using the corresponding log-normal distribution (see [1],[5],[18]). We assume here that the necessary parameters of the log-normal distributions of the values $\theta(\omega, t+1)$ and

$$p(\omega, t+1)/p(\cdot, t)$$

are previously determined by the generic backtesting technique. For example, for (7) we need to identify a, μ, σ, s. This identification procedure is performed using the available statistics in form of a histogram (see e.g., Figure 2).

Assuming the investment volume decision (11), we immediately get a tuning scheme for the PID gains $K(\cdot)$. Combining (6) and (11) we finally obtain

$$\begin{aligned} &\chi(K_P(t+1)\Delta g(t) + K_D(t+1)\dot{\Delta} g(t) + \\ &K_I(t+1)\int_{t-T}^{t} h(\tau)\Delta g(\tau)d\tau) = \\ &p(\cdot, t)\exp\left[\ln\left(\frac{p(\omega, t+1)}{p(\cdot, t)}\right) + \ln\theta(\cdot, t) - \right. \\ &\left.\ln\theta(\omega, t+1) + \ln v(t)\right]. \end{aligned} \tag{12}$$

Since the investment decision $\Delta I(t+1)$ is designed by the PID rule (1)-(2), the obtained formulae (12) constitutes a specific tuning restriction for the PID gains $K(\cdot)$. This novel tuning approach involves the advanced statistical (log-normal) analysis of the financial market behaviour.

IV. Optimization Approaches to the PID Gains Tuning

A credible anticipating strategy for the behaviour of stock markets (using a certain amount of historical data) constitutes the main conceptual problem of the algorithmic trading. In the framework of the proposed PID trading algorithm (1)-(3), we have to incorporate into the PID methodology an

additional optimization procedure for a "best choice" of the PID gains. In this section, we describe two advanced optimization approaches to the optimal PID gains tuning. The resulting PID based strategies (1)-(3) with an optimal gain selection are next called optimal PID trading algorithms. Let us note that the "advanced" character of the optimal tuning mentioned above corresponds to the model-free structure of the proposed PID trading scheme. The model-free nature of an optimal PID implementation (1)-(3) also indicates the usability of the modern ML and RL approaches for optimal PID gains selection problems. Let us refer to [12],[13],[26] for some technical results and novel ideas.

A. Data Driven Gains Optimization

In this section we use a conventional regression based optimization framework applied to the historical stock data. Our aim is to combine the least square optimization with the generic Forward Testing (FT). Note that the FT technique is a common methodology of the modern AT (see e.g., [23],[26]). It includes the so called In-Sample and Out-of-Sample data subsets of the initially given historical data set. The In-Sample data set is part of historical data on which the optimization is performed. The subset of historical data that has been reserved for a possible validation of the optimized trading algorithm is known as an Out-of-Sample data set.

We now choose a number $M_1 \in \mathbb{N}$ as a cardinal number of the In-Sample set and consider the corresponding investment volumes:

$$v^j(t+1),\ j=1,...,M_1-1.$$

Let M_2 denotes a cardinal number of the Out-of-Sample sat such that the complete historical data set under consideration has

$$M := M_1 + M_2$$

elements. Using the given (nonlinear) structure of the PID trading algorithm (1)-(3), we next introduce the following main optimization problem

$$\begin{aligned} &J_1(K(t+1)) := \sum_{j=1}^{M_1} \big(\chi(\delta I(t+1)) - \\ &v^j(t+1)p(\cdot,t+1)\big)^2 \to \min_{K(t+1)} \\ &\text{subject to } (1)-(3),(12) \end{aligned} \tag{13}$$

Evidently, (13) constitutes a specific nonlinear regression determined on the In-Sample data set. Note that problem (13) involves the previously obtained statistical characterisation (12). This statistical result for the PID gains constitutes a natural restriction in the minimization problem (13). The optimization problem (13) can be solved by some known numerical methods (see e.g., [3],[21],[29] and references therein). It finally leads to the optimal PID gains

$$K^{opt}(t+1) := \{K_P^{opt}(t+1),\ K_D^{opt}(t+1),\ K_I^{opt}(t+1)\}.$$

The Out-of-Sample data set can next be used for the validation procedure of optimal solution $K(t+1)$ obtained using the In-Sample set. This validation is based on the comparison of two values of the same objective functional from (13), namely the In-Sample optimal value

$$J_1(K^{opt}(t+1))$$

and the following Out-of-Sample optimal value

$$\begin{aligned} J_2(K^{opt}(t+1)) := \sum_{j=1}^{M_2} \big(\chi(\delta I^{opt}(t+1)) - \\ v^j(t+1)p(\cdot,t+1)\big)^2. \end{aligned} \tag{14}$$

Note that the Out-of-Sample optimal investment

$$\delta I^{opt}(t+1))$$

in (14) is calculated using the optimal gains $K^{opt}(t+1)$ (obtained on the In-Sample data set) and the basic expression from (1). Moreover, the set of investment volumes $v^j(t+1)$ in (14) corresponds to the Out-of-Sample set:

$$v^j(t+1),\ j=1,...,M_2-1.$$

The above optimization problem (13) will be considered as a consistent (optimal) PID gains selection procedure if

$$|J_1(K^{opt}(t+1)) - J_2(K^{opt}(t+1))| \le \varepsilon$$

for a sufficiently small prescribed $\varepsilon > 0$. Let us note that an alternative validation procedure for the PID gains optimization can involve the celebrated Monte Carlo approach (see e.g., [22],[37]).

We now apply the validated optimal gains vector

$$K^{opt}(t+1)$$

for defining the deployed (optimal) investment level

$$\Delta I^{opt}(t+1)$$

using the PID algorithm (1)-(3). The resulting optimal investment volume

$$v^{opt}(t+1)$$

for the time instant $(t+1)$ has the following expression:

$$v^{opt}(t+1) := \frac{\Delta I^{opt}(t+1)}{p(\cdot,t+1)}.$$

Note that the complete trading decision at the time instant $(t+1)$ generated by the proposed algorithm can be formalized using the following signal / volume pair:

$$\begin{aligned} &\{\text{trading}-\text{signal},\ \text{trading}-\text{volume}\} := \\ &\{\text{sign}[\Delta I^{opt}(t+1)],\ \ |v^{opt}(t+1)|\}. \end{aligned}$$

Here sign$[\cdot]$ is the signum function. Moreover,

$$\text{trading}-\text{signal} := \text{sign}[\Delta I^{opt}(t+1)]$$

denotes the optimal trading signal and

$$\text{trading}-\text{volume} := |v^{opt}(t+1)|$$

is the absolute value of the optimal investment volume. The above trading signal / trading volume pair definitively determines a $(t+1)$-trading decision of proposed PID trading algorithm. The proposed nonlinear optimization approach (13) implies that the optimal investment decision

$$\Delta I^{opt}(t+1)$$

constitutes an a priori optimized value in the above signal/volume pair. The corresponding optimal volume $v^{opt}(t+1)$ is in fact an a posteriori value that can be computed after the stock prise $p(\cdot,t+1)$ is known.

B. Stochastic Optimal Gains Tuning

The optimal gains selection problem (13) from the previous section constitutes a data driven regression-like approach. This generic optimization approach leads to the constrained nonlinear optimization and is based on the given historical stock market data. The additional FT technique discussed in the previous section is in fact an adequate expert driven clustering of the complete historical data set. Note that this separation of the data set is methodologically similar to the main idea of the celebrated Monte Carlo method (see [14],[22],[37] and references therein). Let us recall that the classic Monte Carlo method contains the so called "training" and "validation" steps (see [14],[17],[42] and references therein).

In this section we will formulate the stochastic optimization problem for an adequate gains selection in the proposed PID trading algorithm (1)-(3). The stochastic programming problem we consider is conceptually different to the regression-like problem (13). It involves the advanced statistical description of stock markets discussed in Section III. Using (11), we next introduce the following minimization problem:

$$\begin{aligned} &J(\omega,K(t+1)) := \sum_{j=1}^{M} \big(\chi(\delta I(t+1))- \\ &\exp\big[\ln\big(\frac{p(\omega,t+1)}{p(\cdot,t)}\big)+\ln\theta(\cdot,t)- \\ &\ln\theta(\omega,t+1)+\ln v(t)\big]\times p(\omega,t+1)\big)^2 \to \min_{K(t+1)} \\ &\text{subject to } (1)-(3),(12) \end{aligned} \tag{15}$$

The above optimization problem constitutes a nonlinear stochastic program (see e.g., [14],[42]). Clearly, problem (15) now contains a probabilistic costs functional

$$J(\omega,K(t+1)).$$

The stochastic program (15) can be interpreted as a two stage or a multi stage problem. However, we consider here the scenarios based Monte Carlo optimization approach (see e.g., [14]). In this section, we study this problem only from a conceptual point of view and discuss the corresponding Monte Carlo sampling solution scheme.

Taking into consideration the mean approach we replace (15) by the following deterministic program:

$$\begin{aligned} &J_E(K(t+1)) := \sum_{j=1}^{M} E_{\rho_1(\cdot),\rho_2(\cdot)}\big(\chi(\delta I(t+1))- \\ &\exp\big[\ln\big(\frac{p(\omega,t+1)}{p(\cdot,t)}\big)+\ln\theta(\cdot,t)- \\ &\ln\theta(\omega,t+1)+\ln v(t)\big]\times p(\omega,t+1)\big)^2 \to \min_{K(t+1)} \\ &\text{subject to } (1)-(3), \\ &\chi(K_P(t+1)\Delta g(t)+K_D(t+1)\dot{\Delta} g(t)+ \\ &K_I(t+1)\int_{t-T}^{t} h(\tau)\Delta g(\tau)d\tau = \\ &E_{\rho_1(\cdot),\rho_2(\cdot)}\big(p(\cdot,t)\exp\big[\ln\big(\frac{p(\omega,t+1)}{p(\cdot,t)}\big)+\ln\theta(\cdot,t)- \\ &\ln\theta(\omega,t+1)+\ln v(t)\big]\big). \end{aligned} \tag{16}$$

Here

$$E_{\rho_1(\cdot),\rho_2(\cdot)}$$

is the mathematical expectation operator with respect to the (joint) pdf for the pair

$$(\theta(\omega,t),\vartheta(\omega,t))$$

of quotients θ and ϑ introduced in Section III. Note that the log-normal characterization of the probability distribution functions $\rho_1(\cdot)$ and $\rho_2(\cdot)$ implies a specific insufficiency of the use a mathematical expectation in (16). The mean value optimization problem (16) does not possesses the necessary robustness property. This fact can imply some losses of deposit in the case problem (16) is directly applied to trading. In this situation one can replace the mean $E_{\rho_1(\cdot),\rho_2(\cdot)}$ in (16) by a known robust statistical characteristic (median and ctr.) associated with the pdf of the pair $(\theta(\omega,t),\vartheta(\omega,t))$.

Let us also note that the initial stochastic problem (15) as well as the auxiliary (deterministic) problem (16) constitute non-data driven optimization. These abstract problems use the model-based approach that involves the log-normal probability distribution functions $\rho_1(\cdot)$ and $\rho_2(\cdot)$. This fact constitutes a conceptual difference of between the data driven PID gains optimization (13)-(14) and the alternative PID tuning strategy based on the stochastic optimization problem (15).

The auxiliary problem (16) provides a possible approach for the numerical treatment of the initial problem (16). Following the Monte Carlo methodology, we define some probabilistic scenarios

$$P(\omega=\omega_i), i=1.,,,N$$

for some probabilities $P(\cdot)$ associated with the realizations of the stochastic variables ("events")

$$\Gamma := \{\omega_1,...,\omega_N\}.$$

The celebrated scenarios based approximating problem for (16) can now be formalized as follows:

$$
\begin{aligned}
&J_E(K(t+1)) := \sum_{j=1}^{M}\sum_{i=1}^{N} P(\omega=\omega_i)\big(\chi(\delta I(t+1))- \\
&\exp\big[\ln\big(\frac{p(\omega_i,t+1)}{p(\cdot,t)}\big)+\ln\theta(\cdot,t)- \\
&\ln\theta(\omega_i,t+1)+\ln v(t)\big]\times p(\omega_i,t+1)\big)^2 \to \min_{K(t+1)} \\
&\text{subject to } (1)-(3), \\
&\chi(K_P(t+1)\Delta g(t)+K_D(t+1)\dot{\Delta}g(t)+ \\
&K_I(t+1)\int_{t-T}^{t} h(\tau)\Delta g(\tau)d\tau = \\
&\sum_{i=1}^{N} P(\omega=\omega_i)\big(p(\cdot,t)\exp\big[\ln\big(\frac{p(\omega_i,t+1)}{p(\cdot,t)}\big)+\ln\theta(\cdot,t)- \\
&\ln\theta(\omega_i,t+1)+\ln v(t)\big]\big).
\end{aligned}
\tag{17}
$$

Evidently, the finite sum in (17)

$$\sum_{i=1}^{N} P(\omega=\omega_i)g(\omega_i,\cdot),$$

where $g(\omega_i,\cdot)$ denotes the corresponding function in (17) for a concrete event ω_i, approximates the mathematical expectation $E_{\rho_1(\cdot),\rho_2(\cdot)}$ in the auxiliary problem (16).

Similar to the FT methodology used in the previous section, we now divide the above N-dimensional event set Γ into two subsets and define a suitable N_1-dimensional training set and an additional N_2-dimensional validation set

$$\Gamma_1 := \{\omega_1,...,\omega_{N_1}\},$$
$$\Gamma_2 := \{\omega_1,...,\omega_{N_2}\}.$$

Here

$$N_1+N_2=N.$$

with

$$N \gg N_1.$$

The above division of the initially given N-dimensional scenarios set Γ makes it possible to consider the basic Monte Carlo approximating problem (17) on the training set Γ_1 and then on the full scenarios set Γ. In the case of an admissible mismatch in the values of objective functionals of problem (17) considered on Γ_1 and on the full set Γ the optimization procedure is successfully completed. Otherwise, one needs to increase the dimensionality of the historical data set and repeat the above training and validation steps in the Monte Carlo framework.

V. Frequency Domain Representation of the PID Trading Algorithm

A formal frequency domain involved description of the feedback based trading approach can be found in [4]. Let us recall that the frequency domain analysis constitutes a generic approach of the classic CE. We refer to [34] for the corresponding mathematical foundations and computational details of the Laplace and Fourier transforms in the CE framework.

In this section, we discuss the frequency domain analysis of the proposed PID trading algorithm (1) - (3). Let us also observe that the above PID trading algorithm can be interpreted as a control design with time delays. Taking into consideration the main profit formulae (3), the linear expression $\delta I(t+1)$ in (1) can be considered as an ARMA-like formal model. Hence the final expression for $\Delta I(t+1)$ in the PID structure (1) constitutes in fact a nonlinear ARMA abstraction. Note that the consideration of the classic PIDD regulator in the frequency domain constitutes a standard approach of the conventional control systems theory. Let us also observe that the discrete time PID control (4) can be naturally interpreted as a delayed proportional control.

We next apply the celebrated Discrete Signal Processing (DSP) approach and interpret the linear expression for $\delta I(t+1)$ in (1) as a linear filter. Application of the Z-transform to the discrete PID (4) implies the following generic form of the causal discrete-time Finite Impulsive Response (FIR) filter (see [35]):

$$
\begin{aligned}
&Y(z) = (K_P(s+1)+K_I(s+1)+K_D(s+1))z+ \\
&[(K_I(s+1)h(s)-K_D(s+1))z^{-1}+ \\
&K_I(s+1)h(s-1)z^{-2}+K_I(s+1)\sum_{\tau=s-T}^{s-2} h(\tau)z^{-\tau}].
\end{aligned}
\tag{18}
$$

Here $Y(\cdot)$ denotes the resulting Z-transform of the specific output signal $\delta I(t+1)$. Note that z is the proper variable in (18) and $s+1$ indicates the current PID gains. We next can easily obtain the Frequency Response (FR) function for the above FIR filter (18). Recall that FR of the filter (18) is a result of a the formal application of the DTFT (Discrete Time Fourier Transform) to the FIR (18). Following [35] we put

$$z \equiv \exp(-j\omega),$$

where j in (18) denotes the imaginary unit. The FR of the PID trading algorithm (1)-(3) can now be written as follows:

$$
\begin{aligned}
&FR(f) = (K_P(s+1)+K_I(s+1)+K_D(s+1))\times \\
&\exp(-jf)+[(K_I(s+1)h(s)- \\
&K_D(s+1))\times\exp(jf)+ \\
&K_I(s+1)h(s-1)\times\exp(2jf)+ \\
&K_I(s+1)\sum_{\tau=s-T}^{s-2} h(\tau)\times\exp(\tau jf)].
\end{aligned}
\tag{19}
$$

Here f denotes the frequency of a dynamic process. Note that the obtained FR expression (19) for the proposed PID trading algorithm makes it possible to apply the frequency domain methodology for the necessary tuning of the PID gains. The historical stock market data can be used here for an optimal selection of the above gains.

Let us give an illustrative computational example of the FR function. Consider an one day spot rate dynamics of the Binance BTC/USDT exchange (Figure 3).

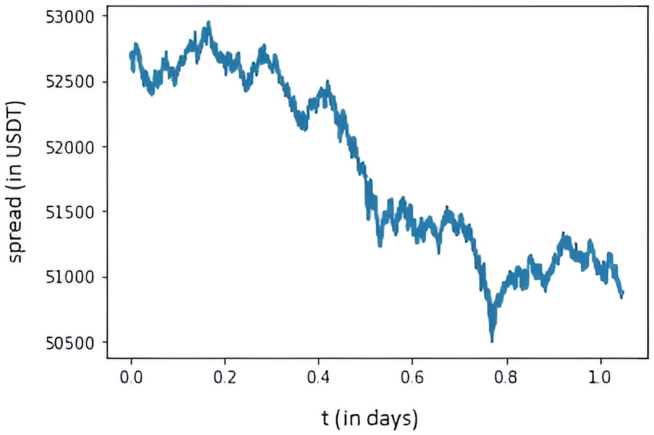


Fig. 3. The one-day spot rate dynamics on the Binance BTC / USDT exchange

The corresponding DTFT of the one day FIR filter dynamics (18), namely, the resulting FR (19) is presented on Figure 4.

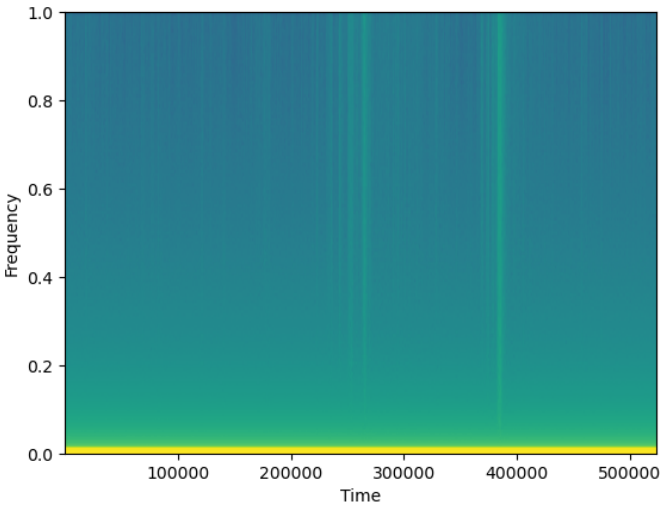


Fig. 4. The FR characteristic of the Binance BTC / USDT spot rate

In fact, the resulting frequency diagram, namely, Figure 4 represents a specific "market energy" distribution by the corresponding frequencies. This energetic interpretation of the presented DTFT is based ob the fundamental Parseval´s theorem in from the classic Fourier analysis (see e.g., [35]). Motivating from this consideration we introduce the following price energy concept associated with the historical stock market data:

$$\mathscr{E} := \sum_{i=1}^{N} FR^2(f),$$

where N corresponds to the dimension of one day data set under consideration and $FR(f)$ is the corresponding Frequency response associated with this data set.

Finally, note that the proposed PID trading algorithm (1)-(3) can also be combined with the several well-known momentum trading strategies (see e.g., [31],[32]). In the framework of a stock market we a are looking for a function

$$S : \Omega \times \mathbb{Z}_+ \to \mathbb{R},$$

the trading strategy. For example, the celebrated fixed-mix strategy $S_{const}(\cdot,\cdot)$ that keeps the value fraction of a risky asset constant has the following easy formalization:

$$S(\omega, t+1) = \frac{c}{p(\omega, t+1)},$$

where c is a given constant. The celebrated (random) trend-following momentum strategy $S(\cdot,\cdot)$ can also be easy represented:

$$S(\omega, t+1) = \mathrm{tr}\{S\}(t+1) + \zeta(\omega, t+1). \tag{20}$$

Here

$$\mathrm{tr}\{S\}(t+1)$$

denotes a trend of the stochastic process $S(\omega, t+1)$. By $\zeta(\cdot,\cdot)$ we denote here the stochastic nois model of the generic market fluctuations. The trend following model (20) provides a future promising tool for an optimal FR based PID gains selection procedure. In fact, a frequency domain analysis provides a natural analytic CE tool for the HFT algorithmic trading.

VI. Prototype Software for Implementation of the PID Trading Algorithm

We now present an application of the developed PID based AT technique to a real-world stock market data. Consider the Binance Bitcoin/USDT futures stock and apply the proposed optimal PID trading algorithm (1)-(3). Consider the BTC/USDT futures price dynamics mentioned above as presented in Figure 5.

Fig. 5. Binance BTC / USD one day price index

We have applied the novel OPID trading algorithm to the above example. The PID gains optimal tuning procedure for

this example was determined as a data driven optimization discussed in Section IV. The corresponding (positive) dynamics of the return is now presented in Figure 6.

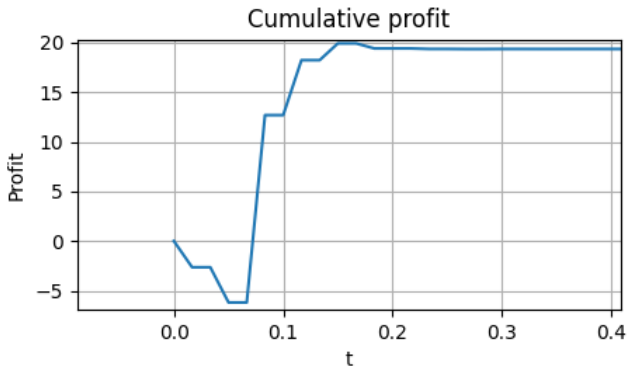


Fig. 6. Binance BTC / USD one day price index

Note that the operation time for every subsequent trading decision of the OPID in this example is timely restricted. The stock market under consideration has an obvious high-frequency behaviour. This fact naturally implies some expected difficulties of the common trading algorithms. In particular, this concerns the widely used moving average trading and the general trend following trading strategies. As one can see, the developed OPID constitutes an adequate approach to the HFT. The time ticks in the above trading example under consideration are significatively dense. The length of the corresponding intervals of trading buckets is equal to 1.728 sec.

Let us also observe the typical "hyperregulation and stabilization" dynamics of the proposed PID trading algorithm (see Figure 3). This dynamic behaviour is a characteristic one for the conventional PID controller (see e.g., [1],[3],[27],[28],[34]). The presented practical example illustrates the implementability of the developed PID trading algorithm AT. Moreover, as we can see the resulting financial behaviour of the return is a profitable behaviour.

VII. Concluding Remarks

In this paper, we developed a novel trading algorithm that involves the model-free PID control methodology and some well established statistical characteristics of the stock market data. The proposed analytic approach can naturally be extended to the real-time multi-asset trading. Moreover, it can also be efficiently combined with some classical trading strategies. The main idea of the proposed PID control approach to the algorithmic trading consists in modern data involved optimization techniques. These advanced optimization procedures determine an optimal calibration (optimal tuning) of the main PID trading parameters, namely, of the PID gains. The resulting optimal PID gains calculated for every stock price tick define a current trading decision. The optimization procedures mentioned above involve the conventional FT techniques in combination with the regression analysis as well as the Monte Carlo method from stochastic optimization.

The given historical stock market data, the statistical properties mentioned above and the data driven optimization techniques are constructively used for an adequate calibration (tuning) of the PID trading algorithm gains. This calibration involves the advanced backtesting procedure. The resulting optimal trading PID strategy generates at every subsequent time instant a profitable decision of the "buy/sell/hold" type for the stock market orders. In fact, the proposed CE like approach to the AT involves a combination of some mathematically rigorous tools, namely, the classic PID control methodology, applied statistics and computational optimization. This interconnected structure of the obtained trading algorithm and the model-free character of the PID scheme indicate the application possibility of the modern ML approaches to the design of the PID type trading algorithms (see e.g., [12],[13],[26]). The above combination of different mathematical tools finally leads to a novel and very promising trading strategy. The resulting implementable PID trading algorithm, the discussed initial software prototypes and the corresponding real-world scenario based simulations extend the family of the feedback based trading algorithms. We also expect a profitable application of the proposed optimized PID trading methodology in the High-Frequency Trading. Note that the developed PID based trading approach can also be involved (as an additional tool) into the several mathematical concepts of the modern financial engineering. For example, it can be considered in the framework of the well established financial time series analysis and Kalman filter techniques (see e.g., [3],[6],[29]). The proposed PID trading algorithm is also compatible with the generic price prediction and trend following techniques (see e.g., [24]).

Finally note that the algorithmic trading strategy proposed in our paper constitutes an initial theoretical development. We are mostly concentrated here on the mathematical and algorithmic aspects of the proposed technique. The prototypes of the financial solutions for the stock market proposed in our contribution need the comprehensive additional analytic development, the corresponding simulations, further backtesting, adequate optimization approaches and practical applications to the real stock markets.

References

[1] V. Azhmyakov, I. Shirokov, Yu Dernov and L. A. Guzman Trujillo, *On the Proportional-Integral-Derivative based trading algorithm under the condition of the log-normal distribution of stock market data*, in: Proceedings of the The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2022), 2022, pp. 17 – 21.

[2] I. Antonioua, V. V. Ivanova, V. V. Ivanov and P. V. Zrelova, *On the log-normal distribution of stock market data*, Physica A, vol. 331, 2004, pp. 617 – 638.

[3] V. Azhmyakov, A Relaxation Based Approach to Optimal Control of Switched Systems, Elsevier, Oxford, UK, 2019.

[4] V. Azhmyakov, J. Pereira Arango, M. Bonilla, R. Juarez del Torro and St. Pickl, *Robust state estimations in controlled ARMA processes with the non-Gaussian noises: applications to the delayed dynamics*, IFAC PapersOnline, vol. 54, 2021, pp. 334 – 339.

[5] V. Azhmyakov, I. Shirokov and L. A. Guzman Trujillo, *Application of a switched PIDD control strategy to the model-free algorithmic trading*, IFAC PapersOnline, vol. 55, 2022, pp. 145 – 150.

[6] O. Bahmani and B. Ford, *Kalman Filter approach to estimate the demand for international reserves*, Applied Economics, 2004, vol. 36, pp. 1655 – 1668.

[7] B. R. Barmish, *On trading of equities: a robust control paradigm*, IFAC Proceedings Volumes, vol. 41, 2008, pp. 1621 -– 1626.

[8] B. R. Barmish and J. A. Primbs, *On market-neutral stock trading arbitrage via linear feedback*, in: Proceedings of the American Control Conference, Montreal, Canada, 2012, pp. 3693 – 3698.

[9] B. R. Barmish and J. A. Primbs, *On a new paradigm for stock trading via a model-free feedback controller*, IEEE Transactions on Automatic Control, vol. 61, 2016, pp. 662 -– 676.

[10] M. H. Baumann, *On stock trading via feedback control when underlying stock returns are discontinuous*, IEEE Transactions on Automatic Control, vol. 62, 2017, pp. 2987 -– 2992.

[11] A. Bemporad, T. Gabbriellini, L. Puglia and L. Bellucci, *Scenario-based stochastic model predictive control for dynamic option hedging*, in: Proceedings of the IEEE Conference on Decision and Control, Atlanta, USA, 2010, pp. 3216 -– 3221.

[12] D. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, Nashua, USA, 2019.

[13] D. Bertsimas and A. W. Lo, *Optimal control of execution costs*, Journal of Financial Markets, vol. 1, 1998, pp. 1 -– 50.

[14] J.R. Birge and F. Louveaux, Introduction to Stochastic Programming, Springer, New York, USA, 2011.

[15] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy, vol. 81, 1973, pp. 637 — 659.

[16] C. Brooks, Introductory Econometcs for Finance, Cambridge University Press, Glasgow, UK, 2015.

[17] G. Cornuejols and J. Pena, R. Tutuncu, Optimization Methods in Finance, Cambridge University Press, Cambridge, UK, 2018.

[18] E. L. Crow and K. Shimizu (Eds.), Lognormal Distributions, Theory and Applications, Marcel Dekker, Inc., New York, 1988.

[19] S. Formentin, F. Previdi, G. Maroni and C. Cantaro, *Stock trading via feedback control: an extremum seeking approach*, in: Proceedings of the Mediterranean Conference on Control and Automation, Zadar, Croatia, 2018, pp. 523 – 528.

[20] R.G. Gallager, Stochastic Processes, Cambridge University Press, NY, USA, 2013.

[21] P.E. Gill, W. Murray and M.H. Wright, Practical Optimization, Academic Press, New York, USA, 1981.

[22] C. Hammel and W. B. Paul, *Monte Carlo simulations of a trader-based market model*, Physica A, vol. 313, 2002, pp. 640 – 650.

[23] T. Hens and M. O. Rieger, Financial Economics, Springer, Berlin, Germany, 2010.

[24] S. Huang, *Online option price forecasting by using unscented Kalman filters and support vector machines*, Journal of Expert Systems with Applications, vol. 34 , 2008, pp. 2819 -– 2825.

[25] P.J. Huber and E.M. Ronchetti, Robust Statistics, Wiley, New York, USA, 2005.

[26] St. Jansen, Machine Learning for Algorithmic Trading, Packt, Birmingham, UK, 2020.

[27] A. Issidori, Nonlinear Control Systems, Springer, London, UK, 1995.

[28] H. K. Khalil, Nonlinear Control, Pearson, Boston, USA, 2015.

[29] F. L. Lewis, Optimal Estimation, Wiley, New York, USA, 1986.

[30] J, Liu and S.J. Wright, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM Journal on Optimization, vol. 25, 2015, pp. 351 – 376.

[31] S. Malekpour, J. A. Primbs and B. R. Barmish, *On stock trading using a PI controller in an idealized market: the robust positive expectation property*, in: Proceedings of the IEEE Conference on Decision and Control, Florence, Italy, 2013, pp. 1210 – 1216.

[32] R. Michaud, Efficient Asset Management, Oxford University Press, UK, 2008.

[33] A. Nemirovski, A. Juditsky, G. Lan and A. Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM Journal on Optimization, vol. 19, 2009, pp. 1574 – 1609.

[34] A. Poznyak, Advanced Mathematical Tools for Automatic Control Engineers: Deterministic Technique, Elsevier, NY, USA, 2008.

[35] A. Poznyak, Advanced Mathematical Tools for Automatic Control Engineers: Stochastic Tools, Elsevier, NY, USA, 2009.

[36] J. Prakash and K. Srinivasan, *Design of nonlinear PID controller and nonlinear model predictive controller for a continuous stirred tank reactor*, ISA Transactions, 2009, vol. 48, pp. 273 – 282.

[37] R.Y. Rubinstein, Simulation and the Monte Carlo Method, John Wiley Inc., New York, USA, 1981. 1981).

[38] M.B. Rudoy and C.E. Rohrs, *A dynamic programming approach to two-stage mean-variance portfolio selection in cointegrated vector autoregressive systems*, in: Proceedings of the IEEE Conference on Decision and Control, Cancun, Mexico, 2008, pp. 4280 – 4285.

[39] A. Shapiro and T. Homem-de-Mello, *On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs*, SIAM Journal on Optimization vol. 11, 2000, pp. 70–86.

[40] S. Taylor, Modeling Financial Time Series, Wiley, Chichester, UK, 1986.

[41] N. Vo and R. Slepaczuk, *Applying hybrid ARIMA-SGARCH in algorithmic investment strategies on S&P500 index*, Entropy, vol. 24, 2022.

[42] R.J-B Wets, *Stochastic programming*. in: G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, Eds., Optimization, Handbooks in Operations Research and Management Science vol. 1, North–Holland, Amsterdam, Netherlands, 1990.

[43] S.A. Zenios, Financial Optimization, Cambridge University Press, Cambridge, UK, 1993.

[44] W.T. Ziemba and R.G. Vickson, Stochastic Optimization Models in Finance, Academic Press, New York, USA, 1975.

Toward Leveraging Code Generation Architectures for the Creation of Evolvable Documents and Runtime Artifacts

Herwig Mannaert, Gilles Oorts, Jan Verelst

Normalized Systems Institute
University of Antwerp, Belgium
Email: herwig.mannaert@uantwerp.be

Koen De Cock and Jeroen Faes

Research and Development
NSX bv, Belgium
Email: koen.de.cock@nsx.normalizedsystems.org

***Abstract*—Many organizations are often required to produce large amounts of documents in various versions and variants. Though many solutions for document management and creation exist, the streamlined automatic generation of modular and evolvable documents remains challenging. The challenges are to some extent similar to the automatic generation of modular and evolvable software, which has been the subject of previous work on metaprogramming. In this contribution, a proof of concept architecture is presented to generate modular documents from runtime information systems through the use of a reduced runtime version of this metaprogramming environment. The configuration and integration of this expansion kernel into regular applications, and its use to generate some basic administrative document sources are explained. Based on this architecture, several use case scenarios are explored to generate other types of documents and artifacts using live runtime data.**

***Index Terms*—Evolvability; Normalized Systems Theory; Metaprogramming; Document Creation; Single Sourcing**

I. Introduction

This paper extends a previous paper, which was originally presented at the Thirteenth International Conference on Pervasive Patterns and Applications (PATTERNS) 2021 [1].

Organizations are often required to produce large amounts of versions and variants of certain documents. While they have traditionally focused their efforts into streamlining technical product documentation [2], they are now also looking to build business value by creating personalized customer-faced documents [3]. At the same time, current information systems are producing massive amounts of relatively simple documents, e.g., invoices and timesheets, based on corporate data.

The streamlined and possibly automatic generation of such documents has been associated with concepts like modularization and single sourcing [2], both reminiscent of similar techniques used in the creation of software. Similar to software development, dealing with versions and variants of modules may lead to so-called ripple effects, i.e., changes in certain versions or variants of modules may require changes in other modules. To facilitate such often necessary changes and to provide a desired level of evolvability, document structures need to be designed that deplete this rippling of changes [4]. Moreover, the use of parameter data during the instantiation of document variants seems similar to the inner workings of code generation environments.

In our previous work, we have presented a meta-circular implementation of a metaprogramming environment [5], and have argued that this architecture enables a scalable collaboration between various metaprogramming projects featuring different meta-models [6][7]. In this contribution, we investigate the use of a reduced version or runtime kernel of this code generation environment within the generated software applications. More specifically, we present a prototype implementation for the creation of evolvable artifacts, such as documents, where runtime data of the generated software application is used to instantiate the artifacts. The presented approach is neither confined to a specific domain, nor to a specialized type of software. However, the implementation is currently limited to the generation of relatively simple and straightforward documents.

The remainder of this paper is structured as follows. In Section II, we briefly discuss some aspects and terminology related to the creation and single sourcing of documents, an important class of artifacts created by information systems at runtime. In Section III-A, we explain the basic concept of Normalized Systems Theory with regard to the design of evolvable artifacts. Section III-B recapitulates the architecture of our meta-circular code generation environment, and explains that this expansion of source code artifacts is not limited to programming code. Section IV presents how a runtime kernel of this generation environment can be configured, and integrated into regular applications, to instantiate and expand runtime artifacts such as documents based on live data. Section V explores some use case scenarios to leverage this runtime expansion environment for the automated generation of various document types. Finally, we present some reflections and conclusions in Section VI, and discuss future work.

II. Modular and Evolvable Document Creation

While organizations have traditionally focused their efforts in document management into streamlining product documentation [2], there is a widespread belief that personalized customer-faced documents can build business value by enhancing customer loyalty [3]. However, repurposing internal documents to be used for online purposes, such as sales, marketing, product documentation and customer support has proven to

be diffcult. Moreover, it is hard to find any best practices or repeatable models developed that address this challenge [2]. In this section, we briefly discuss some techniques and issues regarding the creation of evolvable documents.

A. Document Creation and Single Sourcing

A successful approach to handle any complex system or problem is modularization [8][9]. An example of such an approach in the area of document management is *Component Content Management (CCM)*, defined as *a set of methodologies, processes, and technologies that rely on the principles of reuse, granularity, and structure to allow writers to author, review, and repurpose organizational content as small components* [2]. One of the fundamental ideas of component content management is the separation of content and layout [10]. The granularity of a component in CCM is defined by the smallest unit of usable information [11]. Several standards exist that define practical and technical implementation guidelines for creating modular and reusable content. According to Andersen and Batova [2], the most widely implemented standard is the *Darwin Information Typing Architecture (DITA)*.

Originally regarded as the broader discipline of CCM in the early 2000s, *single sourcing* has been defined as one of the fundamental aspects of CCM concerned with the design and production of modular, structured content. An elaborate description of single sourcing and its concepts, advantages, methodology, guidelines and practical examples, can be found in [12]. There are three fundamental aspects to single sourcing. First, content is made *reusable* by separating content from format. A second aspect is *modular writing*. Content is written in stand-alone modules instead of whole documents. This allows content to be assembled into documents from *singular source files that contain unique content*, the third aspect of single sourcing. Besides *assembling* the content modules into documents, i.e., combining source files in a hierarchical and sequential way with a distinct combination of audience, purpose and format, the modules need to be *linked*, i.e., connected to make them into coherent documents.

Enabling the content creators to focus on the actual substance of documents instead of having to deal with layout and publishing technologies, should lead to various advantages: saving time and money, improving document usability, and increasing team synergy [12]. Single sourcing recognizes two types of document creation. *Repurposing* entails merely reusing content modules for a different output format. *Re-assembly* on the other hand, is a more impactful way of reusing modules to develop documents for different purposes or audiences. Contrary to repurposing, re-assembly also includes changing the sequence of modules, the conditional inclusion, and the hierarchical level of inclusion.

B. Modular and Parametrized Document Generation

The emergence of concepts like modularization, CCM, and single sourcing regarding the management of certain classes of documents, e.g., technical documentation or personalized documents, is highly reminiscent of similar concepts in software codebases. Indeed, software developers have been striving for decades to modularize codebases, to separate concerns into singular source files, and to assemble source code modules into software applications, in a continuous effort — or quest — to reuse and repurpose these source modules. The component in CCM, defined by the smallest unit of usable information [11], seems to be consistent with the concept of a module in a software source base.

Both documents and software source bases can have successive *versions* in time that contain additions, corrections or omissions to its content, and can be branched into concurrent *variants* when variations in content and/or purpose occur. Just like in software, dealing with versions and variants of a document requires the design of document structures to provide a desired level of evolvability. Evolvable documents are documents that do not hinder or limit the application of changes made to their structure or content. They are free from ripple effects that would cause changes to the documents to be highly difficult and costly [4].

Documents, such as technical and/or personalized documents, can have many concurrent variants. In technical documents, these variants range from the variation or even conditional presence of entire technical descriptions and procedures due to differences in the components of various installations, to simple parameter values like the serial number, color, or location of the documented installation or product. But short and simple documents, like letters, invoices or timesheets, can also be considered to have many variants due to different parameter values. This aspect of parameter-based or *model-based instantiation of document variants*, is highly reminiscent of environments for *code generation in software development*.

III. Expansion of Evolvable Modular Structures

In this section, we discuss the expansion and assembly of evolvable modular structures. We introduce *Normalized Systems Theory (NST)* as a theoretical basis to design information systems —and conceptually other kinds of modular structures— with higher levels of evolvability, and its realization in a framework to generate and assemble programming code, and possibly other types of source artifacts.

A. Normalized Systems Theory and Evolvable Structures

NST was proposed to provide an ex-ante proven approach to build evolvable software [13][14][15]. It is theoretically founded on the concept of *systems theoretic stability*, a well-known systems property demanding that a bounded input should result in a bounded output. In the context of information systems, this implies that a bounded set of changes should only result in a bounded impact to the software. This implies that the impact of changes to an information system should only depend on the size of the changes to be performed, and not on the size of the system to which they are applied. Changes causing an impact dependent on the size of the

system are called *combinatorial effects*, and considered to be a major factor limiting the evolvability of information systems. The theory prescribes a set of theorems, and formally proves that any violation of any of the following *theorems* will result in combinatorial effects (thereby hampering evolvability) [13][14][15]:

- *Separation of Concerns*
- *Action Version Transparency*
- *Data Version Transparency*
- *Separation of States*

Applying the theorems in practice results in very fine-grained modular structures in software applications, which are in general difficult to achieve by manual programming. Therefore, the theory also proposes a set of patterns to generate significant parts of software systems that comply with these theorems. More specifically, NST proposes five *elements* that serve as design patterns for information systems [14][15]:

- *data element*
- *action element*
- *workflow element*
- *connector element*
- *trigger element*

Based on these elements, NST software is generated in a relatively straightforward way. Due to this simple and deterministic nature of the code generation mechanism, i.e., instantiating parametrized copies, it is referred to as *NS expansion* and the generators creating the individual coding artifacts are called *NS expanders*. This generated code can be complemented with custom code or *craftings* at well specified places (anchors) within the skeletons or boiler plate code. This results in the structural separation of four dimensions of variability [15][7]:

1) *Mirrors* representing data and flow models, using standard techniques like Entity Relationship Diagram (ERD) and Business Process Model and Notation (BPMN).
2) *Skeletons* expanded by instantiating the parametrized templates of the various element patterns.
3) *Utilities* corresponding to the various technology frameworks that take care of the cross-cutting concerns.
4) *Craftings* or custom code to add non-standard functionality that is not provided by the skeletons.

It has been extensively argued that the design theorems and structures of NST are applicable to all hierarchical modular architectures that exhibit cross-cutting concerns [16]. More specifically related to documents, the software theorems and element patterns of NST are very similar to the principles of CCM that rely on reuse and fine-grained modular structures to allow writers to author, review, and repurpose organizational content as small components, and to the concept of single sourcing, demanding the separation of content and layout. Moreover, it has been shown that the application of NST to the design of evolvable document management systems leads to architectures that are in accordance with the principles of CCM and single sourcing [4][17][18].

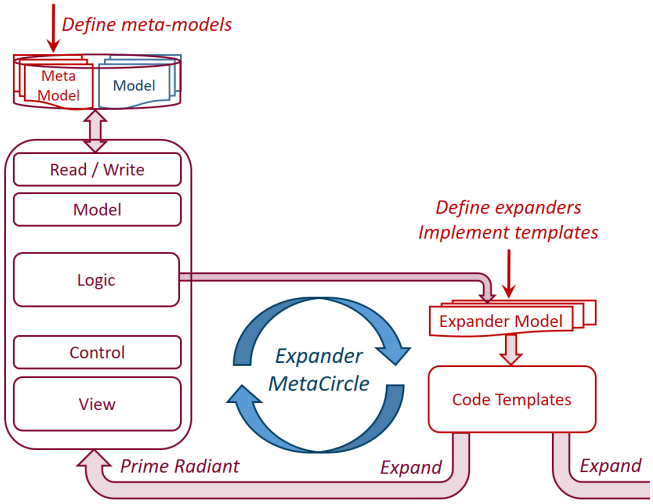


Figure 1. The meta-circular architecture for NS expanders and meta-application.

B. Meta-Circular Code Generation or Artifact Expansion

NST has been realized in software through a code generation environment to instantiate instances of the various elements or design patterns. Due to the simple and deterministic nature of this code generation, i.e., instantiating parametrized copies, it is referred to as *NS expansion*. We have also argued that nearly all metaprogramming or code generation environments exhibit a rather similar and straightforward internal structure [6][7], distinguishing:

- *model files* containing the model parameters.
- *reader classes* to read the model parameter files.
- *model classes* to represent the model parameters.
- *control classes* to select and invoke the generator classes.
- *generator classes* instantiating the source templates, and feeding the model parameters to the source templates.
- *source templates* containing the parametrized code.

As the NST metaprogramming environment was developed for the creation of web information systems, it has always included the generation of various building blocks, e.g., reader and model classes, which are similar to those of the code generation environment itself. This has made it possible to merge those generated code modules with the corresponding code generation modules, thereby evolving the metaprogramming environment into a meta-circular architecture [5]. This meta-circular architecture, described in [5][6][7], is schematically represented in Figure 1 and entails several advantages. First, this architecture enables the *regeneration of the metaprogramming code itself*, thereby avoiding the growing burden of maintaining the often complex meta-code, such as adapting it to new technologies. Second, it allows for a structural decoupling between the two sides of the code generation transformation, i.e., the domain models and the code generating templates. This also removes the need for contributors to get acquainted with the — basically non-existing —internal code structure of the metaprogramming environment, as additional

expanders with corresponding coding templates can be defined and activated using a declarative control mechanism.

We have argued in previous work that the meta-circular architecture presented in Figure 1 enables what we call two-sided collaboration [7]. By allowing the definition of interfaces at both ends of the code generation transformation, different groups of developers can collaborate simultaneously on the models and on the implementation templates. At the template side of the interface, templates are not limited to programming code containing statements and commands of programming languages like *Java* or *JavaScript*. Analogous to programming code, templates may be defined that contain commands and settings of markup languages and/or typesetting systems like *Markdown* or LaTeX, leading to the generation of non-code artifacts such as document sources. At the meta-model side of the interface, the definition of additional meta-models is not limited to programming structures either. Instead of representing software components, data and task elements, the meta-models may just as well correspond to hierarchical document modules such as chapters and sections, that will be realized by the corresponding commands and settings of the typesetting systems like *Markdown* or LaTeX.

Therefore, any model representing parameter data and/or small content components, may serve as a meta-model and drive the expansion or instantiation of the document. The meta-circular architecture does not require any explicit programming to support the new model entities representing the document. As we have seen, the various classes corresponding to the new model entities (XML readers and writers, model classes, control and generator classes) will be automatically generated. Analogous to the meta-circular *Prime Radiant*, the meta-application of the NST metaprogramming environment, a generated application that is based on such a new meta-model and allows the expansion of artifacts based on this meta-model, was originally called a *Secondary Radiant*. However, as the need for the creation of documents is nearly omnipresent in information systems, it is desirable to enable the creation of artifacts such as documents for all information systems and their models. Therefore, an architecture was designed to support the expansion of artifacts in every regular normalized systems application, treating every model as a meta-model. We refer to this architecture as the *Runtime Radiant*.

It should be noted that a meta-circular code generation environment is not necessarily complex. For instance, we have presented in detail how an elementary meta-circular code generation environment can be bootstrapped [19]. During this bootstrapping process, it is also shown how the generated code itself becomes able to generate other artifacts.

IV. Toward a Systematic Implementation for the Runtime Expansion of Documents

In this section, we present a prototype implementation of the *Runtime Radiant* architecture for the expansion of parametrized documents using the NST meta-circular code generation or metaprogramming environment.

A. Document Creation and Information Systems

As explained in Section II, an interesting duality exists between information systems and document creation. Information systems often support the creation of simple documents, such as invoices or timesheets, incorporating data that is entered and managed within the information system. At the same time, the streamlined creation of large amounts of document variants, for instance in the case of technical product documentation, requires some tooling to specify and manage the various parameters that drive the creation of the document variants. In other words, information systems often create documents, and document creation systems usually require a supporting information system.

For the prototype implementation targeted at the creation of documents using the NST meta-circular code generation environment, we have opted for the first scenario. The streamlined creation of variants of complex documents would require the definition of an elaborate meta-model describing the structure and domain parameters of the documents. The creation of such a model is out of scope of this contribution. However, as the creation of such a meta-model corresponds essentially to the design of a suitable data model or ontology, we feel that this does not pose a significant technological risk. Therefore, we decided to explore the generation of rather simple documents based on common data entities like invoices or timesheets. Nevertheless, this prototype implementation of the *Runtime Radiant* architecture does address a possible and important technological hurdle. As these documents need to incorporate runtime data from the live information systems, e.g., the actual details of the various invoices. Therefore, this proof of concept validates the expansion of artifacts based on runtime data from *any* information system expanded by the NST metaprogramming environment. In this way, the implementation can also serve as a validation for the expansion of other source artifacts based on live runtime data of information systems, such as marketing emails or sensor configuration files.

B. Declarative Control and Runtime Expansion

Consider two typical samples of a simplified data model for an administrative information system as presented in Figure 2.

- An *invoice* with some attributes, e.g., an invoice number and reference, containing a reference to a client, and consisting of several invoice lines.
- A *timesheet* with some attributes, e.g., the month and employee, containing a reference to a project, and consisting of several timesheet entries.

These data entities are expanded into *data elements*, collections of software classes as described in [7], by the NST metaprogramming environment, and incorporated in a web-based information system. The expanded data elements or collections of classes include:

- Reader and writer classes to read and write the XML data files, e.g., *InvoiceXmlReader* and *InvoiceXmlWriter*, *TimesheetXmlReader* and *TimesheetXmlWriter*.

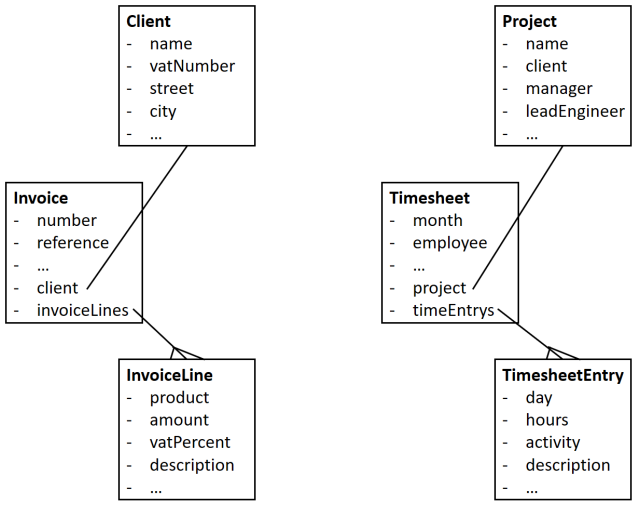


Figure 2. Samples of a simplified data model for an administrative system.

```
<expander name="TexInvoiceExpander"
  xmlns="http://normalizedsystems.org/expander">
  <packageName>net.palver.tex.invoice</packageName>
  <layerType name="ROOT"/>
  <technology name="COMMON"/>
  <sourceType name="TEX"/>
  <elementTypeName>Invoice</elementTypeName>
  <artifact>Invoice-$invoice.number$.tex</artifact>
  <artifactPath>$expansion.directory$/
                $artifactSubFolders$</artifactPath>
  <isApplicable>true</isApplicable>
  <active value="true"/>
</expander>
```

Figure 3. Declaration document of a *TexInvoiceExpander*.

- Model classes to represent and transfer the various entities, and to make them available as an object graph, e.g., *InvoiceDetails* and *InvoiceComposite*, *TimesheetDetails* and *TimesheetComposite*.
- View and control classes to perform *CRUDS (create, retrieve, update, delete, search)* operations in a generated table-based user interface.

In the same way that the instances of the NST meta-model data elements are read and made available as an object graph at the time of code generation, the instances of the data elements represented in Figure 2 can be made available as an object graph at runtime in a generated information system. Incorporating the core templating engine of the NST metaprogramming environment, as described in the following subsection, allows to evaluate the various attributes of the administrative data entities using *Object-Graph Navigation Language (OGNL)* expressions, and to feed them to the text templates, e.g., in LaTeX, that are used to create the invoice and timesheet documents.

As explained in [7], the expansion of artifacts, e.g., source code or document files, is based on a generic *ArtifactExpander* that uses declarative control to evaluate the model parameters and insert them into the source templates. Every individual expander generating a source artifact is defined in an *Expander*

```
<mapping
 xmlns="https://schemas.normalizedsystems.org/
               xsd/expanders/2021/0/0/mapping">
 <value name="info" eval="invoice.info"/>
 <value name="number" eval="invoice.number"/>
 <value name="client" eval="invoice.client.name"/>
 <value name="vatNr" eval="invoice.client.vatNr"/>
 <value name="street" eval="invoice.client.street"/>
 <value name="city" eval="invoice.client.city"/>
 <value name="isForeign"
  eval="!invoice.client.country.equals('Belgium')"/>
 <list name="invoiceLines"
       eval="invoice.invoiceLines"
       param="invoiceLine">
  <value name="info" eval="invoiceLine.info"/>
  <value name="product" eval="invoiceLine.product"/>
  <value name="amount" eval="invoiceLine.amount"/>
 </list>
</mapping>
```

Figure 4. Mapping document of a *TexInvoiceExpander*.

XML document. An example of the definition of such an individual expander to expand a LaTeX source file for an invoice is shown in Figure 3. It is quite similar to the declaration of an expander creating a Java source file during code generation, but has a `tex` source type, and uses for instance the runtime invoice number to construct the filename.

The evaluation of the various instance parameters or attributes is based on OGNL expressions [20] and defined in a separate *ExpanderMapping* XML document. This ensures the separation of content from format, as required by [10] to have reusable and evolvable documents. An example of the definition of such an individual mapping document for the invoice creation is shown in Figure 4. Besides simple OGNL expressions, it allows to evaluate logical expressions, e.g., whether the invoice client is foreign for VAT purposes, and to define lists of linked objects, e.g., invoice lines, and to loop through these lists and access the attribute values of the members of the list.

The values as defined in the expander mapping document are passed to the LaTeX templates. As described in [6], the NST environment uses the *StringTemplate (ST)* engine library [21]. This library supports the creation of a modular document structure by providing *subtemplate include* statements, enabling the document designers to adhere to the principles of single sourcing [12]. For instance, we share the declaration of various LaTeX packages and the definition of some basic commands through the use of the subtemplates `<basePackages()>` and `<baseCommands()>`. And the various invoice lines of an invoice (or timesheet entries of a timesheet) are created by instantiating a corresponding subtemplate for every list item through `<invoiceLines:invoiceTableLine()>` (or `<timesheetEntrys:timesheetTableLine()>`).

C. Integration in Normalized Systems Applications

A reduced version or kernel of the NST metaprogramming environment, incorporating the core templating engine, was packaged into a runtime installation that can be integrated

into every expanded normalized systems application. This runtime installation provides an interface to invoke expanders by passing instances of a tree of plain data classes, i.e., so-called *DTOs (Data Transfer Objects)*. The runtime kernel will *expand an artifact for every data tree instance that is passed to an expander (template)*.

The expansion of an artifact by the runtime kernel of the NST metaprogramming environment is schematically represented in Figure 5, both for expanding a Java software class for an *Invoice data entity* (left side), and for expanding a Latex document for an *actual instance of an invoice* (right side). In both cases, the expansion basically intertwines —the process is similar to a mathematical convolution— an instance of a data entity (tree) with a source template.

- An instance of an agent interface class for the *data element Invoice*, i.e., `InvoiceAgentIf`, is expanded by combining the instance data tree (including linked elements like *fields*) for the *data element* named `Invoice` with the template of the Java interface class `AgentIf`.
- An instance of a Latex invoice for the *invoice Inv-001*, i.e., `Inv-001TexInvoice`, is expanded by combining the instance data tree (including linked elements like *invoice lines*) for the *invoice* named `Inv-001` with the template of the Latex document `TexInvoice`.

Though the runtime expansion kernel can be used by any application that is able to provide (trees of) data instance classes, the integration in a normalized systems application is particularly straightforward and economical. First, the required (trees of) data transfer classes can be generated automatically by the NST metaprogramming environment. Second, an option has been introduced to generate a specific implementation class for a task element, that automatically feeds an instance of such a data class tree to a defined set of expanders. This expander set merely needs to be defined in a configuration file in the expansion resource. Such a resource is a standard library archive that contains the actual expanders, i.e., the triplets consisting of an expander definition, an OGNL mapping file, and a template. This task element can be invoked in a line of code, by a button in the user interface, but can also be embedded in a workflow driven by an NST flow element. In the latter case, the expansion of the document can be automatically performed for every instance of the target data entity for which a dedicated field has a specific value.

The integration of this runtime expansion kernel, the so-called *Runtime Radiant*, has been tested in a normalized systems business application that included (a more elaborated version of) the data elements represented in Figure 2. Based on live data from this operational production environment, many hundreds of LaTeX sources for invoices and timesheets were successfully generated through the use of the expander declarations and OGNL parameter evaluations as presented above. The processing of the Latex sources into *PDF* documents was integrated by embedding an additional task element in the workflow of the NST flow element.

It is clear that this expansion architecture allows information systems to create other type of document source artifacts based on live runtime data. Indeed, as the NST expansion environment is agnostic with respect to the source type, e.g., able to create LaTeX source documents in exactly the same way as Java source files, the generation of other types of document source modules is basically reduced to creating other types of templates and defining them in expander declarations. It is worth noting that such generated documents can have multiple remote impacts in our networked world. For instance, HTML documents generated by the runtime expansion kernel can be sent out immediately to large amounts of users through email or direct messaging. Or generated XML configuration documents can be uploaded automatically to remote Internet of Things (IoT) sensors or controllers, resulting in the flexible configuration of those devices.

V. Exploring Some Target Document Types for the Runtime Artifact Expansion

In this section, we explore the feasibility of several use case scenarios to apply and leverage the proposed architecture for the runtime expansion of various document types.

A. Typical Information Systems Documents

Though the automated creation of various small documents and reports is nearly omnipresent in contemporary information systems, we nevertheless believe that the additional possibilities and/or advantages of the proposed runtime expansion architecture can be significant.

First, the runtime expansion architecture brings a built-in capability for document creation to every piece of data of every information system that is able to provide (trees of) data transfer objects. Such a capability could enable end-users to define and author various classes of documents to be created, as opposed to the current situation where they have to either rely on specific types of documents that are supported by the information system, or use a dedicated application for document authoring and creation. While the former case could be limiting, the latter case could lead to consistency issues.

Second, the deep integration of the document expansion kernel into the information system, enables easy and structured access to possibly large areas of the data model. Consider for instance the generation of *Curriculum Vitae (CV)* documents. Though an enormous amount of applications exist to manage and generate employee CVs, the use of the runtime expansion architecture could give the document creation engine immediate access to a wide variety of personnel data, such as project involvement, completed training programs, and even performance appraisals. Moreover, it would intrinsically support the introduction of dedicated query logic, like the selection of the most relevant experiences for a target customer, based for instance on the industry sector and the project team size of past project involvements.

Third, the proposed architecture could provide a unified way to create a whole range of document artifacts. Besides

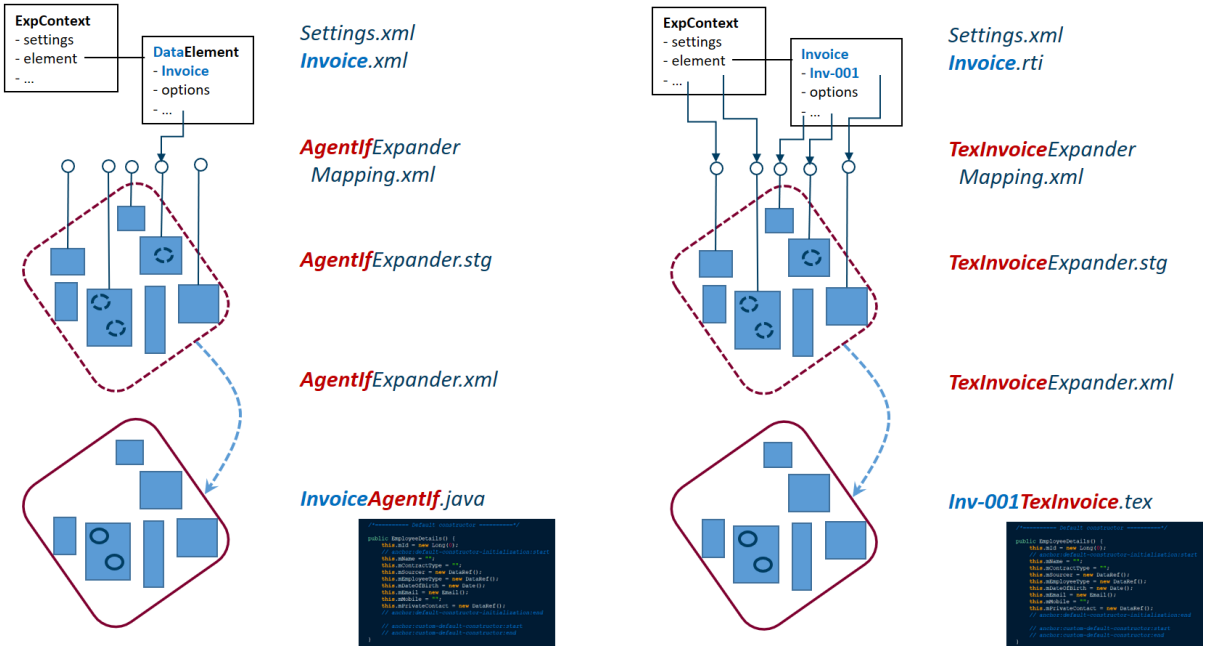


Figure 5. Schematic representation of an expansion of software class (left) and text document (right).

traditional documents like invoices and timesheets, we mention XML messages for financial reporting purposes or e-commerce transactions, and HTML documents for announcements and marketing campaigns. While these types of documents or messages are typically generated by other subsystems, a unified architecture could not only entail economies of scale, but would also contribute to consistency and single sourcing [2]. For instance, both the PDF invoice and the XML version based on the *Universal Business Language (UBL)* standard [22], could be based on the same data entries. Moreover, the proposed architecture could not only establish single sourcing across different types of content, but extend it to integration architectures between different information systems. Revisiting the CV example, one could for instance imagine to exchange CV data through machine readable structured document formats such as XML or JSON.

B. More Complex Hierarchical Documents

The advanced capabilities of a code generation environment with respect to hierarchical modular structures could be beneficial for certain types of more complex documents. The built-in hierarchical structure of the metaprogramming environment could enable the automated generation of documents with an elaborate and/or complex hierarchical structure, that require both different versions over time, and multiple variations for a given version, for instance depending on the profile of the target audience. In accordance with the detailed analysis of such documents with a complex hierarchical structure in [4], we mention the following examples.

- Detailed technical documentation of modular artifacts such as heat transformers or compressors, where different instances of the artifact may contain different (versions of) individual parts, and different descriptions and/or sections are appropriate depending on the target audience.
- Detailed security rules and guidelines for technical installations such as power plants, where guidelines often depend on regulations and legislation that evolve over time, and where specific instructions need to be tailored to individual installations and staff profiles.
- Elaborate assessment documents as required for self-assessments and external audits, where (versions of) various descriptions and/or sections, and even whether they need to be present at all, are in general dependent on the target audience and/or the scope of the assessment. A case related to such self-assessment documents for the evaluation of study programs is described in detail in [18].

For these types of documents, the proposed runtime expansion architecture could be an enabler for the structured hierarchical approach toward document creation described in [17][18].

C. Toward Integrating Multimedia Content

The integration of the document creation kernel in any information system could also facilitate the introduction of multimedia content. Having for instance such a system for the generation of CVs, would simply require the definition of one or more video clip data element(s) to enable the uploading of multiple types of personal message clips into the system. Such clips could both be bundled or integrated with the CV, and integrated through embedded links to the clips.

As discussed in a case study on a video learning channel [23], multimedia content itself could be hierarchically structured with different versions and variations. One could imagine

a runtime expansion environment to aggregate video lectures based on an hierarchical modular structure, selecting the appropriate content modules, language tracks and background themes. Provided the integration of some technical utilities, the runtime expansion system could combine the various modules and aspects into an integrated video lecture. Once again, this could foster collaboration between lecturers and content providers, while respecting the concept of single sourcing.

VI. Conclusion and Future Work

Many organizations are often required to produce large amounts of versions and variants of documents in areas like technical documentation and accreditation. At the same time, corporate information systems are producing massive amounts of relatively simple documents based on corporate data. The streamlined and possibly automatic generation of such documents has been associated with concepts like modularization and single sourcing, which are similar to techniques used in code generation software. As in software, dealing simultaneously with different versions and variants requires the design of document structures to deplete the rippling of changes in order to provide a desired level of evolvability.

In our previous work, we have presented a meta-circular implementation of a metaprogramming environment, and have argued that this architecture can be used for code generation based on different and even newly defined meta-models. In this contribution, we have investigated the use of this code generation environment within the generated information systems at runtime. More specifically, we have explored the creation of evolvable artifacts, such as simple administrative documents, where live runtime data of the generated software application is used to instantiate the artifacts.

To this purpose, we have packaged a reduced version or runtime kernel of the NST metaprogramming environment, and presented an architecture to integrate this expansion kernel into the runtime environment of every expanded information system. More specifically, we have shown how (trees of) instances of data transfer objects, containing runtime data, can be passed to source templates to generate artifacts such as document sources. The usage of this runtime expansion kernel in information systems to generate sources for administrative documents based on templates, only requires declarative definitions and OGNL evaluation mappings, and does not imply any dedicated software programming.

This paper provides different contributions. First, we validate that it is possible to use the NST metaprogramming environment to create another type of source code artifacts, e.g., document sources in some typesetting system. Moreover, we have explained that this implementation adheres to several fundamental concepts regarding modular and evolvable document creation, like CCM and single sourcing. Second, we validate that we can integrate a reduced kernel version of the NST metaprogramming environment into a runtime information system expanded by the NST metaprogramming environment, and to generate source artifacts from live data within this running information system.

Next to these contributions, it is clear that this paper is also subject to a number of limitations. First, we have only demonstrated the integration of the runtime expansion kernel into information systems generated by the NST metaprogramming environment. Second, the generated documents are quite simple, and in line with documents that are currently generated by mainstream information systems.

Nevertheless, this explorative proof of concept can be seen as an executable architecture, and we are planning future work to extend both the scope and the use of this environment in several ways. First, we intend to generate more types of document sources. We mention for instance XML-UBL documents for electronic invoices, *Fast Healthcare Interoperability Resources (FIHR)* for healthcare information exchange, and various XML documents to automatically configure sensors and controllers in energy monitoring and management systems. Second, we plan to significantly increase the use of this document generation environment. Besides the current production use for invoices and timesheets, we are considering its use for car policy documents, CVs, training certificates, meeting notes, et cetera. This usage would not only be confined to applications expanded by our NST metaprogramming environment, but will also be made available to other applications. Third, we intend to start addressing more complex hierarchical documents such as technical manuals and audit reports. To facilitate this use case, we have introduced in our tooling the possibility to define meta-models as ontologies, independent from our web application models. This will enable users to create document generation applications based on the structural models of their manual or reports, without having to deal with the technicalities of the traditional NST web information systems.

References

[1] H. Mannaert, G. Oorts, K. De Cock, and S. Gallant, “Exploring the use of code generation patterns for the creation of evolvable documents and runtime artifacts,” in Proceedings of the Thirteenth International Conference on Pervasive Patterns and Applications (PATTERNS), April 2021, pp. 17–22.

[2] R. Andersen and T. Batova, “The current state of component content management: An integrative literature review,” IEEE Transactions on Professional Communication, vol. 58, no. 3, 2015, pp. 247–270.

[3] S. Abel and R. A. Bailie, The Language of Content Strategy. Laguna Hills, CA, USA: XML Press, 2014.

[4] G. Oorts, Design of modular structures for evolvable and versatile document management based on normalized systems theory. Antwerp, Belgium: University of Antwerp, 2019.

[5] H. Mannaert, K. De Cock, and P. Uhnák, “On the realization of meta-circular code generation: The case of the normalized systems expanders,” in Proceedings of the Fourteenth International Conference on Software Engineering Advances (ICSEA), November 2019, pp. 171–176.

[6] H. Mannaert, C. McGroarty, K. De Cock, and S. Gallant, “Integrating two metaprogramming environments : an explorative case study,” in Proceedings of the Fifteenth International Conference on Software Engineering Advances (ICSEA), October 2020, pp. 166–172.

[7] H. Mannaert, K. De Cock, P. Uhnák, and J. Verelst, “On the realization of meta-circular code generation and two-sided collaborative metaprogramming,” International journal on advances in software, vol. 13, no. 3-4, 2020, pp. 149–159.

[8] H. Simon, The Sciences of the Artificial. MIT Press, 1996.

[9] C. Y. Baldwin and K. B. Clark, Design Rules: The Power of Modularity. Cambridge, MA, USA: MIT Press, 2000.

[10] D. Clark, "Content management and the separation of presentation and content," Technical Communication Quarterly, vol. 17, no. 1, 2007, pp. 35–60.

[11] F. Sapienza, "A rhetorical approach to single-sourcing via intertextuality," Technical Communication Quarterly, vol. 16, no. 1, 2007, pp. 83–101.

[12] K. Ament, Single Sourcing: Building Modular Documentation. Norwich, NY, USA: William Andrew Publishing, 2003.

[13] 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, 2011, pp. 1210–1222, special Issue on Software Evolution, Adaptability and Variability.

[14] ——, "Towards evolvable software architectures based on systems theoretic stability," Software: Practice and Experience, vol. 42, no. 1, 2012, pp. 89–116.

[15] H. Mannaert, J. Verelst, and P. De Bruyn, Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design. Koppa, 2016.

[16] H. Mannaert, P. De Bruyn, and J. Verelst, "On the interconnection of cross-cutting concerns within hierarchical modular architectures," IEEE Transactions on Engineering Management, vol. 69, no. 6, 2022, pp. 3276–3291.

[17] G. Oorts, H. Mannaert, and P. De Bruyn, "Exploring design aspects of modular and evolvable document management," in Proceedings of the Seventh Enterprise Engineering Working Conference (EEWC), May 2017, pp. 126–140.

[18] G. Oorts, H. Mannaert, and I. Franquet, "Toward evolvable document management for study programs based on modular aggregation patterns," in Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications (PATTERNS), February 2017, pp. 34–39.

[19] H. Mannaert and K. De Cock, "Bootstrapping meta-circular and autogenous code generation," in Proceedings of the Seventeenth International Conference on Software Engineering Advances (ICSEA), October 2022, pp. 87–92.

[20] "OGNL," URL: https://en.wikipedia.org/wiki/OGNL, 2022, [accessed: 2022-06-15].

[21] "StringTemplate," URL: https://www.stringtemplate.org/, 2022, [accessed: 2022-06-15].

[22] "Universal Business Language (UBL)," URL: https://en.wikipedia.org/wiki/Universal_Business_Language, 2023, [accessed: 2023-02-01].

[23] H. Mannaert, I. Franquet, C. Lippens, and K. Martens, "Exploring various aspects of a video learning channel : the educational case study of eclips," International Journal On Advances in Intelligent Systems, vol. 12, no. 13-4, 2019, pp. 169–176.