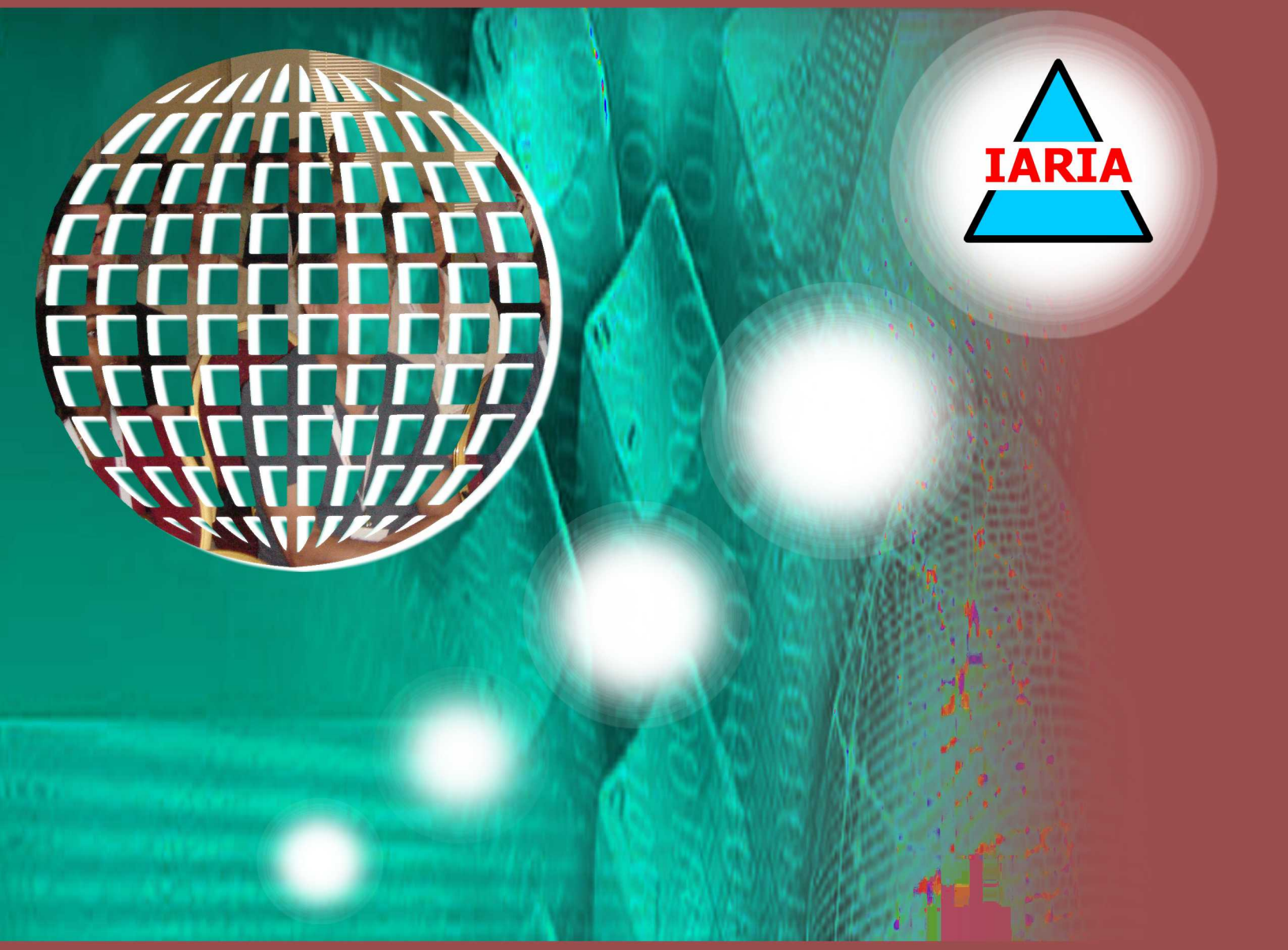
International Journal on Advances in Software



2022 vol. 15 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 15, Number 1 & 2, 2022

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

Zhensheng Guo, Siemens AG, Germany
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

International Journal on Advances in Software
Volume 15, Numbers 1 & 2, 2022

CONTENTS

Environment Code-First Framework: Provisioning Scientific Computational Environments Using the Infrastructure-as-Code Approach

Daniel Adorno Gomes
University of Trás-os-Montes and Alto Douro
Vila Real, Portugal
www.utad.pt
e-mail: adornogomes@gmail.com

Pedro Mestre and Carlos Serôdio
Centro Algoritmi
University of Minho, Guimarães, Portugal and
CITAB - Centre for the Research and Technology of Agro-Environmental and Biological Sciences
University of Trás-os-Montes e Alto Douro
www.utad.pt
e-mail: pmestre@utad.pt, cserodio@utad.pt

Abstract— **Nowadays, computational resources are vital practically in all areas of science. At the same time, science's dependence on computing is pointed to by experts as one of the main causes of the reproducibility crisis. Many factors have contributed to the low reproducibility of scientific research. They are related to the cultural aspects of the scientific software's development, the behavior of the scientist-developer, and technical issues. Based on these factors, the authors presented the Environment Code-First (ECF) framework to guide researchers on increasing the reproducibility of their works by developing computational environments that can be easily recreated without manual intervention. The framework's foundation is the Infrastructure-as-Code approach, and it intends to permit other researchers to recreate an environment only by executing a script. A real case is presented, demonstrating the provision of a bioinformatics environment by using the Prokaryotic Genomics and Comparative Genomics Analysis Pipeline (PGCGAP) protocol, and the ECF framework. The paper shows a comparison between these two methods in terms of time-consumption, manual intervention, platform-agnosticism, and portability. The tests perfomed on three different machines demonstrated that there are many benefits on using the ECF framework such as independency of platform, total portability, and practically any manual intervention. Of course, there is a cost, and it is related to the hard work on developing the code that generates the environment. Another point that needs to be highlighted is the time spent and efforts on achieving the necessary knowledge to create those programs.**

Keywords-computational environment; infrastructure-as-code; open science; computer programming; containerization; virtualization; reproducible research.

I. INTRODUCTION

The use of computing is essential in all sciences. Many areas such as biology, physics, and chemistry are dependent on simulations to perform experiments in silico. It is faster and cheaper to execute simulations than conduct actual experiments. In other situations, it is impossible to conduct an experiment without computational resources, for example, when it is necessary to process and analyze a large amount of data in a short space of time. Over the past 70 years, research methods have become more and more sustained by computational means. Nowadays, this dependency is pointed by specialists as one of the main factors responsible for the crisis of scientific reproducibility [1][2].

Reproducibility is one of the most important pillars of science, along with transparency and openness [3]. It is a crucial element to recreate and extend the research work developed by others. And this practice is critical to keep science evolving.

From an economic point of view, many losses on investment have been reported in the last few years related to the reproducibility of scientific research. In [4], authors report that, in the United States, around 50% of the total amount invested every year in biomedical research, is supporting scientific studies that other researchers cannot reproduce. Recently the European Commission reported that the losses related to low reproducibility on clinical trials are estimated at 28 billion USD per year [5].

Besides the crisis in science related to reproducibility [6], there is pressure from funding institutions and publishers on authors to adopt open science principles like Open Reproducible Research (ORR) [7]. This means to provide access to the resources (e.g., data, source code, documentation) used to generate the results reported in their publications [8].

These factors increase the need to improve reproducibility and transparency in science, especially in research where computation has an important role [9]. In a survey published by Nature with 1576 researchers, Baker asked them which kind of factors contribute to the irreproducibility. More than 40% of respondents reported computational issues, such as unavailability of computational methods, code, and data [10].

Despite the absence of code and data being the most common issues on reproducibility, it is essential to highlight the problems related to the computational environments that support the research. Incompatibility of operating systems, different versions of the same compiler or interpreter, lack of software packages and libraries, the dependency of libraries of a specific software platform, are all issues related to the environment used to develop scientific applications. Accordingly with Boettiger in [11], these kind of computational issues related to the environment can be classified as dependency hell or code rot.

It is critical for the reproducibility of any research work that made use of computational resources to provide the

environment besides the code, the data, and the documentation [12]. As Donoho wrote in [13]: “An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.”

As the way to share scientific knowledge is changing from traditional articles to this new concept based on Open Reproducible Research, it is necessary to change the way scientific applications are developed, focusing not only on generating results but on being reproducible and useful to the wider scientific community, as well [14].

Today, we can count on technologies (e.g., virtual machines, containers and cloud computing), and new technical approaches that permit the treatment of the computational infrastructure that supports scientific research as a software system, a method also known as Infrastructure-as-Code (IaC). Using Infrastructure-as-Code, we can provide the infrastructure programmatically, having many benefits such as treating the infrastructure like a computer system, versioning it, and avoiding typical issues like dependency hell [15]. However, there are many cultural barriers related to scientific software development that contributes to irreproducibility, despite the technical and technological resources available. They are related to the purpose of the scientific applications, the behavior of scientists when developing software, the lack of use appropriate software engineering practices, among other factors.

Considering the issues, particularities and characteristics related to the development of scientific applications, and having the IaC approach as a technical foundation, the authors present in this paper the Environment Code-First Framework. The goal of the framework is to help to increase reproducibility by reducing the time and the efforts when reproducing specific research, mitigating issues related to the availability of the computational environment created and used by the researchers. It guides the scientists on developing the infrastructure's code to make the computational environment available before they start to develop the scientific application in itself. In the end, the environment will be a deliverable as the others objects used and produced by the research, being stored and accessible in the same repository (e.g., Git or Github) with them. That means other researchers will have access to environment code, application code, data, and documentation, all together, avoiding reproducible issues.

The framework defines an architecture based on virtual machines and containers, the two technologies most used by the scientific community in the last fifteen years to create self-contained computational environments. The innovation, in this case, is in the fact that the framework combines the two technologies instead of using them in an isolated way. This approach permits developing more homogeneous environments independent of hardware and software platforms.

Besides increasing reproducibility and transparency of new scientific research works, this proposal can be helpful in other aspects like education and dissemination of the open science principles. It can help experient and future researchers understand and prepare to produce executable papers and migrate old research to this new approach. Also, it can help bring down cultural barriers that impede the advance of the open science philosophy, such as authors' hesitation to share their work to avoid publishing erroneous papers.

The rest of the paper is organized as follows: Section 2 presents some typical technical issues on reproducible research, related to scientific computational environments. Section 3 presents some cultural aspects of the scientific community that difficults the increasement of reproducibility. Section 4 presents some related work on IaC. Section 5 makes an explanation about what is this new approach called Infrastructure-as-Code. Section 6 presents the Environment Code-First framework. Section 7 describes a case study of a recent bioinformatics pipeline implementation, and compares it to the ECF framework implementation. Section 8 presents the discussion. Finally, in Section 9 it is presented the conclusion.

II. TECHNICAL ISSUES

In this section, the authors present some typical issues on reproducible research, related to computational resources, which have been faced by researchers.

Collberg et al. show in [16] the technical issues that researchers have to deal when trying to reproduce results published by others. In general, considerable effort is needed to recreate the original computational environment and achieve similar results. The authors analyzed 613 executable papers. Only 123 applications were correctly compiled. Of them, 102 ran with success. Unavailability of a specific version of a software component and missing third-party packages or libraries are the reasons that caused 36% of the failed builds.

In [17], Glatard et al. expose how complex pipelines that are composed by several parts of software from different sources, can have unexpected outputs or a failed execution of the entire workflow, due minor changes introduced in the computational environments, for example, when a updated is applied in the operating system.

In [11] the author describes a typical issue called "code rot". It is a kind of issue that affects the results of the original code due to updates applied in the software environment to fix bugs, add new features, or deprecate old ones. All the software that composes the environment like the operating system, the development language, and the libraries, can be affected by an update generating different results from the original. Also, the author describes another problem known as dependency hell. It occurs when installed software packages have dependencies on specific versions of other software packages. It also includes platform-specific dependencies that are related to a software development platform. The most common types of dependency hell are DLL hell, JAR hell [18].

Ince et al. reported in [19] problems related to differences between the published and the reproduced results using the same source code of computer programs that were implemented and executed with success by non-original researchers. The issue, in this case, occurred when the programs were deployed using hardware and software

configurations that diverged from the original. As a solution, the authors suggested that the source code of the programs should be made available along with documentation describing the hardware and software environment on which the program was developed and should be executed.

In [20] Ben Marwick highliths how important is, in archaeology, the simulation studies executed by computationally-intensive analysis that use mathematical functions based on single-precision floating-point arithmetic. In this case, the issues are related to the variation of the results when executed accross different opearting systems. Also, the author describes the high difficulty in maintaining an environment reproducibly, even when using only one machine, due to automatic updates that software components sofer considering an extended period.

III. Scientific Applications and Cultural Barriers

Besides the technical issues reported in the last topic, there are other factors related to scientific applications' purposes and the behaviors of the scientists that have impacted the reproducibility of scientific works.

Scientific applications are a special category of software generally developed to support comprehending a particular scientific domain that would be impossible to perform without computational resources. Also, the scientific application development process differs significantly from the development of traditional information systems.

As shown in [21], most of the habits and behaviors of the scientists, when the subject is software development, had been adopted during more than last 60 years. Over this period, a culture had emerged and disseminated, creating the role of the scientist end-user developer, as defined by the authors. In this role, the scientist is responsible for planning, designing, developing, testing, and using the results generated by the application. Most of the time, the scientist works alone, using a PC, focusing entirely on the scientific problem.

In [20], Ben Marwick highlights most scientists' primary computational environment is a PC (i.e., desktop or laptop) using one of these three operating systems, Microsoft Windows, MacOS, or Linux.

Hannay et al. performed a survey with almost 2000 respondents to investigate how scientists develop and use scientific software. On the computational environment, 48.5% of the scientists reported that they use exclusively a desktop or a laptop when working on their scientific applications [22].

As reported in [23], a survey performed with 60 scientific software developers, around 80% of the respondents develop their applications alone.

Related to the software engineering principles and practices, generally, most researchers do not test, document, or release their applications [24]. In [25], a systematic mapping study on using software engineering practices for scientific application development, those facts are reinforced. The study points to that self-education is the most common way adopted by researchers when learning about software development. Also, it highlights that the absence of training is one reason for the low level of knowledge of the researchers on software engineering practices and their benefits.

IV. What Kind of Solutions Has Been Reported?

In this section, the authors present a set of works related to IaC that had applied this practice in different areas of the software industry and scientific applications.

Boettiger in [11] discusses the issues of reproducible research with a focus on the computational environment that supported the research. He describes the main issues that impede the successful execution and extension of the code by other researchers. Besides, he makes a review of some approaches used in IaC such as containerization and virtualization. The author analyzes in-depth containerization based on Docker technology, showing the advantages of this approach, such as portability, reusability, versioning, and cross-platform, and how it can help provision computational environments for scientific research.

In [26] the authors present a study on the benefits that could be brought by the adoption of cloud computing and virtualization techniques in scientific applications. They discuss a cultural problem that avoids using virtualization and cloud resources due to the idea that virtualization techniques hurt the results of the scientific applications in comparison with the execution of physical machines. Also, they explore the feasibility of the IaC approach to meet the requirements of computer science and the main issues that need to be addressed by cloud actors to provide the conditions necessary to obtain the maximum benefit from this type of infrastructure.

In [27], the authors present the main characteristics of cloud computing technology, highlighting those that can help in the development of more robust applications based on aspects such as scalability, resilience, fault tolerance, and security. Besides, they discuss the low cost of adopting cloud computing and show how to transition from traditional biomedical computing workflows to cloud computing environments.

In [28], the authors discuss the complexity on creating scientific computing environments. They discuss common issues in the scientific community like the inability of the scientists on setting up isolated and uniform computational environments. Abscense of best practices, software redundance problems, platform dependency are some of the situations described by the authors. To address these kind of issues, they present means to use DevOps concepts, practices and tools to improve the provision of computational environments and reduce the complexity. The use of virtualization, containerization and configuration management are some of the resources used by DevOps engineers suggested in this paper.

Howe discusses in [29] the challenges of provisioning computational environments for scientific research projects through virtualization on cloud computing platforms. The author presents how a complete working environment of specific research containing dataset, software, notes, logs, and scripts, can be included in a virtual machine. Also, he presents a discussion on the advantages and disadvantages of

the adoption of this approach, and how it can help to increase reproducibility.

Cacho and Taghva [30] present the main difficulties researchers face in reproducing research in the Computer Science field. Some of the problems they highlighted include missing original data, issues with the version of the data, deprecated dependencies, unavailable source code, and missing documentation. They provide a solution for reproducible research based on containerization. A real experiment is used to demonstrate the solution using the application OCRSpell. The authors make the provision of a Docker container that embeds the application by creating an image, uploading it, and making it available to other researchers that want to run the OCRSpell. From this image a researcher just needs to download the image and run the container.

Ben Marwick in [20] demonstrates in a practical way how to produce an executable paper relating principles of reproducible research to DevOps practices and tools. The author describes the efforts to create a publication of archaeological research using resources like Docker, R programming language, Git, and Linux operating system. Also, the author explains each tool used to develop the paper and exposes the reasons that motivate the use of each resource.

V. What Is Infrastructure-as-Code?

Infrastructure-as-Code (IaC) is the management and provisioning of IT infrastructure using source code rather than manual processes. It automates the provisioning of infrastructure and eliminates the need to provision and manage servers, operating systems, database connections, storage, and other infrastructure elements manually, avoiding mistakes.

The infrastructure is treated like a software system, which means development tools and agile practices such as Test-Driven Development (TDD), Continuous Integration (CI), and Continuous Delivery (CD) can be used to improve the quality [31]. Programmatically defining our infrastructure means that our environments will be more consistent and reliable, and identical every time. Manual provisioning generally has diverse interpretations of the same instructions, resulting in different configurations [32].

IaC is based on a few practices as follows [15]:

- All the provisions and configurations related to the infrastructure are defined in executable files, such as shell scripts. The actions that need to be applied in the infrastructure like installing a database, increasing the memory of a server, and even creating a new server, are executed from these files.
- The scripts contain the commands that make the maintenance of the infrastructure, and the documentation of the systems and processes.
- The scripts are the source-code that represents the infrastructure. They need to be kept in a version control system like Git or Github.
- Even in infrastructure source-code, tests are critical to finding errors. Continuous integration pipelines can be set up to test and guarantee the quality of the code, supporting practices like continuous delivery and deployment, which can help decrease the downtime of the systems on upgrades or fixes.

There are many benefits to adopting IaC due to the following characteristics of this approach [33][34]:

- Repeatability: having the infrastructure defined as code ensures that we can recreate it as many times as needed, getting the same result.
- Automation: creating and configuring the infrastructure from executable scripts are tasks that can be automated in addition to mitigating manual intervention and avoiding human mistakes.
- Agility: using resources such as source code management systems and version control systems to store the infrastructure code permits us to apply changes anytime, responding to defects and business demands faster because we can always backward the infrastructure to a known state.
- Scalability: the combination of repeatability and automation allows us to increase our infrastructure in an easy and fast way.
- Consistency: repeatability and automation also guarantee that we will always have the same environment, as defined in the source code.
- Disaster recovery: as we have all the infrastructure defined as code, in case of a catastrophic event where we lose all the environment, it will be easy to recover and recreate it from our source code repository.

The most common issues that IaC addresses in the software industry are related to environment similarity and scalability. Regarding environment similarity, the IaC is helping companies increase the similarities between development, testing, and production environments and ensuring applications have the same behavior in any of them. It also avoids the differences that generally are present when creating the infrastructure by manual intervention. In terms of scalability, the approach is being used by companies that need a high level of dynamism in their infrastructure, like e-commerce. For example, IaC permits rapidly increasing the number of servers when the sales volume is growing and reducing them when the sales are decreasing, which is essential to control the costs [34].

The IaC approach appeared due to the evolution and growth of cloud computing demands. In general, this new approach is related to cloud-based environments, but, it can be used in on-premises infrastructure, as well. Even, it can be applied to isolated machines [35].

There are different ways and tools to implement IaC, depending on the need. The most common tools used in scientific environments are that related to the provision of virtual machines or containers, which embed the software environment, the dataset, and the source-code that composes a specific experiment [11][29].

Using IaC, researchers can write and execute code to define, deploy, update, and destroy the necessary infrastructure for their experiments. This code will be stored

in a version control system, making the experiment reproducible as many times as needed, by the originals and other researchers [10] [11].

VI. THE ENVIRONMENT CODE-FIRST FRAMEWORK (ECF)

Based on the findings presented before, there are three main issues related to software environments that support scientific research and directly impact on their irreproducibility. The first is the absence of proper documentation providing a step-by-step on how to reproduce the environment, including all software like libraries, packages, compilers, interpreters, databases, their versions, and how to install and configure them. The second is the dependency of a specific platform of software or hardware due to the use of the researchers made of their personal computers. Creating a computational environment in a specific machine with a specific operating system forces us to use the same platform to reproduce the same environment. The last one is the dependency hell. When a computational environment is built on a specific machine by a researcher in a manual way, it is being created with many dependencies on libraries, packages, and software versions that will exist only on that machine. It is almost impossible to reproduce the same environment on other machines, especially by other researchers different from the original. Producing unique computational environments in software and configuration is an issue called snowflake servers [15] or snowflake systems [34].

As discussed earlier, these issues have their origins in the following root causes arising from the cultural and behavioral aspects of the scientist developer:

- The use of the researchers made of their personal computers;
- Most of the time, the scientist end-user developer works alone or in small teams;
- The scientist end-user developer does not produce documentation;
- Researchers are not interested in software engineering best practices because their focus is on the research. The applications developed by the scientist end-user developer are just tools that help him obtain and process the data they need to analyze.

Also, some of the root causes identified have characteristics that contribute to the manual intervention of the researchers when creating the computational environments that support their research work. The practice of the installation and configuration of the environment in a manual way can be considered another root cause that aggravates the second and the third issues mentioned before.

In this paper, a framework is being proposed to mitigate these issues and to conduct the researchers to create self-contained computational environments more reproducible, isolated, portable, and independent of any platform of software and hardware.

The framework shall drive the development of an environment that is:

- Independent of hardware and software platform, regarding operating systems;
- Ready to run on-premise, on a PC or more powerful servers, or even in the cloud;
- Fully provisioned programmatically, with no installations and configurations manually performed, using a repository (e.g., Github or GitLab) to store the source code of the environment;
- Dynamic in terms of software resources, allowing the addition or remotion of them from the environment's source code at any time, and in a fast way;
- Storable in small files, in Megabyte order, not Gigabyte order, without the need for exhaustive downloads;
- Quickly reproducible and ready to use in the order of minutes.

The framework has two parts. The first part determines the architecture of the computational environment, and the second is a guide that defines a step-by-step procedure that must be followed by the researcher for the development and maintenance of the infrastructure.

A. The ECF Architecture

The main goal of the architecture defined by the ECF is to create a homogeneous environment independent of the hardware and software platforms used by the researcher. For example, a team of five researchers using various types of PCs (e.g., notebooks and desktops) with different operating systems would still have the same environment in all machines when following the ECF framework.

In the last fifteen years, two main technical approaches had been used by the scientific community to create self-contained computational environments proper to reproducibility. The first one are the virtual machines (VMs) and the other are the containers. Both permit us to create packaged computing environments composed of many IT elements (e.g., CPU, memory, and storage) available in file format. When executed inside the host machine, both isolate their environments from the rest of the system. But, while the VMs offer complete isolation from the host operating system, the containers offer lightweight isolation. It occurs due to the use of containers made of the host machine's resources, while VMs have their operating system, CPU, memory, network interface, and storage. This is an advantage of the VMs in terms of security and a barrier in terms of availability and portability, because of the size of their files. The size of a container image file is generally measured in MB, while that of a VM can take several GB [36]-[38].

Nowadays, there is a growing adoption and use of Linux container technology (e.g., Docker, LXC) compared to virtual machines by the software industry and the scientific community. Many scientific papers have been published presenting executable paper solutions based on containers to help grow reproducibility and transparency in science [11][20][30][39][40]. However, the proposal presented by the most of papers related to this subject is usually to use the containers directly on the host machine. This can be a

challenge in terms of platform because the containers have to be compatible with the operating system [41]. Currently, Docker is considered the de facto standard for containerization [42]. The Linux containers based on this standard, only can run directly on machines that have a Linux distribution as the operating system. To run Docker Linux containers on other operating systems such as Microsoft Windows and MacOS, it is necessary to install and configure a set of software that will adapt them to support Linux containers, usually in a manual way [43].

The ECF defines an architecture composed of two modules to permit researchers to obtain precisely the same computational environment when reproducing a research work. The first, called Container Module (CM), is a Linux container with all the software, libraries, and packages needed to develop and run the scientific application. The second, called Virtual Machine Module (VMM), comprehends a hypervisor, a lightweight virtual machine based on Linux distribution, and a container engine (e.g., Docker). In the development phase, the modules will be developed separately. But, in the execution phase the CM will run inside the VMM, on top of the container engine layer. Practically, the CM will work as another layer of the CMM. As shown in Fig. 1, the layers in green represent the physical machine and the operating system installed on it. The other layers, that appears involved by a dotted line, are all part of the architecture defined by the ECF framework. Both modules have to be provisioned programmatically using IaC resources such as ad-hoc scripts, configuration management tools, server templating tools, orchestration tools, and provisioning tools.

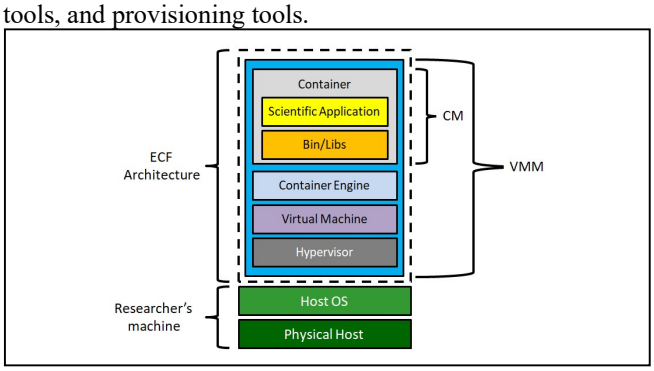


Figure 1. The ECF architecture.

Provisioning the computational environment based on the ECF architecture guarantees the container will always run on the same operating system, independent of the software platform used by the researchers on their physical machine, as shown in the example of Fig. 2.

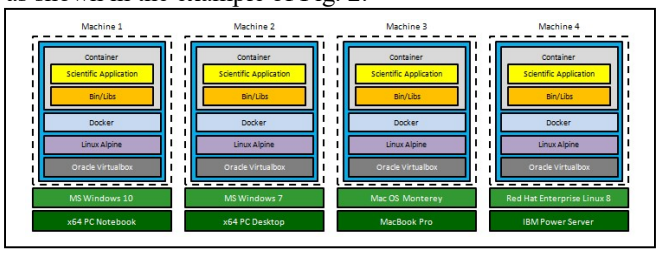


Figure 2. Example of the ECF architecture running on different platforms.

B. The ECF Guidence

The goal of this part of the framework is to guide the researchers on implementing the two modules defined by the ECF architecture, the CM and the VMM. The CM have to be implemented first, as it is one of the four layers that compose the VMM. For both modules, the framework establishes a series of steps that have to be systematically followed by the researchers to get each of them implemented.

1) The CM implementation guide

The implementation of the CM can be summarized in five high-level guidelines:

- Requirements identification;
- Development of the CM source code;
- Source code storage;
- Container image generation;
- Container image storage.

Fig. 3 shows the guidelines in a diagram where we can see in which order researchers have to perform such actions. It is essential to notice the researcher will not perform these actions only once but in a cycled way as many times as needed due to the maintenance of the environment. For example, after the environment is ready and running, if a researcher identify a need to add a new library, it will necessary execute all the steps to get the new version of the container image stored and available for download.

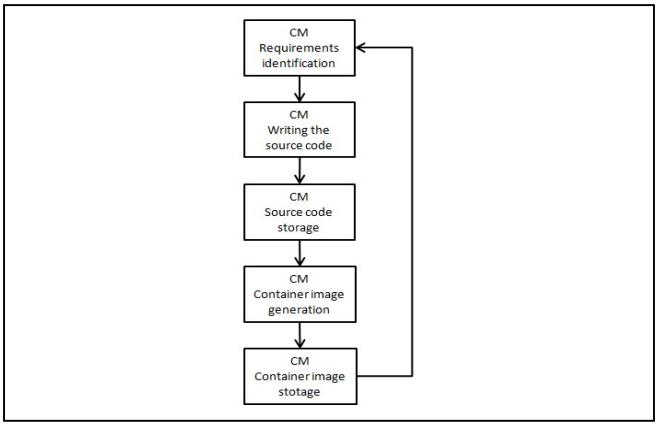


Figure 3. Steps involved in the CM development.

In the requirements identification step, the researcher will identify all the software, libraries, and packages necessary to develop and run the scientific application. The first requirement that has to be defined in this step is the container engine. It is mandatory to know what container engine will be used to build the environment before all the other requirements. The source code the researcher will write to define the installations and configurations to create the environment depends on this definition. It has to follow the patterns and syntax required by the chosen engine. The other requirements can vary from one environment to another, depending on the experiment's needs. The source code files must contain all the instructions and explanations needed to document the commands and configurations specified. The ECF framework defines a form model with a set of questions based on the most common types of software components

used in scientific environments that have to be answered by the researchers when analyzing the requirements to build the CM. Of course, this model must be adapted for each case, including or removing requisites according to the situation. Table I shows the form with the questions defined by the ECF. Besides the column with the question, there are two more columns in the form. One for the answer in itself, and the other two specify the software's version.

TABLE I. CONTAINER MODULE DEVELOPING FORM

Question	Answer	Version
Which container engine will be used?		
Which will be the base image of the containers?		
Which programming languages and compilers will be used?		
Which libraries, packages and third part software have to be installed?		
Which databases will be installed?		
Is it necessary to perform any configuration? In which files?		
Is it necessary to copy any files into the container? Which files?		

The next step consists of developing the container image's source code based on the requisites identified before. After writing the container image's source code, the researcher needs to store it in a version control system like Github and Gitlab. In the following step, it is necessary to compile the source code to generate the image used to create the container that supports the scientific application's development and execution. Also, it is necessary to perform some tests with the image. The last step is to store the image in a container repository (e.g., Docker Hub).

2) The VMM implementation guide

Similarly to the CM, the implementation of the VMM is composed by three high-level guidelines:

- Requirements identification;
- Development of the VMM source code;
- Source code storage.

The sequence the steps have to be executed is shown in Fig. 4. Although the diagram demonstrates that the steps can be performed cyclically, it is not typical for the VMM because it will not change with as much frequency as the CM can change. For example, it will be necessary to change the VMM when we have to increase the quantity of memory or the number of CPUs.

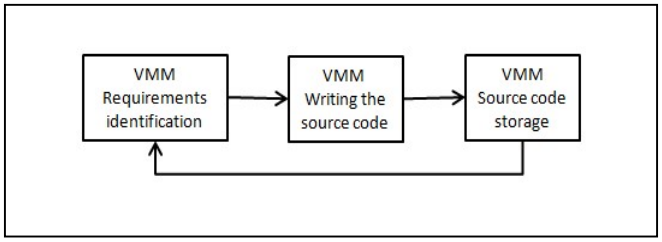


Figure 4. Steps needed to create the VMM.

The VMM is limited to four essential layers: the hypervisor, the virtual machine, the container engine and, the CM in itself. That means there is no need to add or remove any layer. In the requirements identification step, the researcher will have to define which hypervisor will support the virtual machine, how much memory it will use, how many CPUs it will have, and which operating system will be installed. At this point, the container engine is already known, and the CM is ready to use. The other definitions are related with the IaC tools the researcher wants to use to develop the VMM. Also, the ECF defines a form with the main questions to guide the researcher in this phase. It can be visualized in Table II.

TABLE II. VIRTUAL MACHINE MODULE DEVELOPING FORM

Question	Answer	Version
Which hypervisor will be used?		
How much memory will be allocated for the virtual machine?		
How much CPUs will be dedicated to the virtual machine?		
Which operating system will be installed on the virtual machine?		
Which IaC tools will be used to automate the provisioning of the environment?		

At this point, the researcher has all the elements that is necessary to develop the source code of the VMM. The source code must guarantee the hypervisor installation on the physical machine, the provisioning of the virtual machine with the container engine and the CM inside it. The documentation about the actions performed to create the VMM have to be included in the source code files. The source code produced in this phase must be stored in a version control system, but the virtual machine image does not. It will be used only to create an instance of the container image that represents the scientific environment. In this way, the virtual machine image only has to keep stored locally, and it can be destroyed and recreated as many times as needed. During the initialization, the VMM has to check if a new version of the CM is on the image repository. In a positive case, it needs to download the new container image before instantiating it. Once the VMM source code covers all these actions and is already available in a version control system, any researcher can use the produced scripts to recreate an environment.

In Fig. 5, a flowchart shows all the steps involved in running a VMM script on provisioning an entire scientific computational environment.

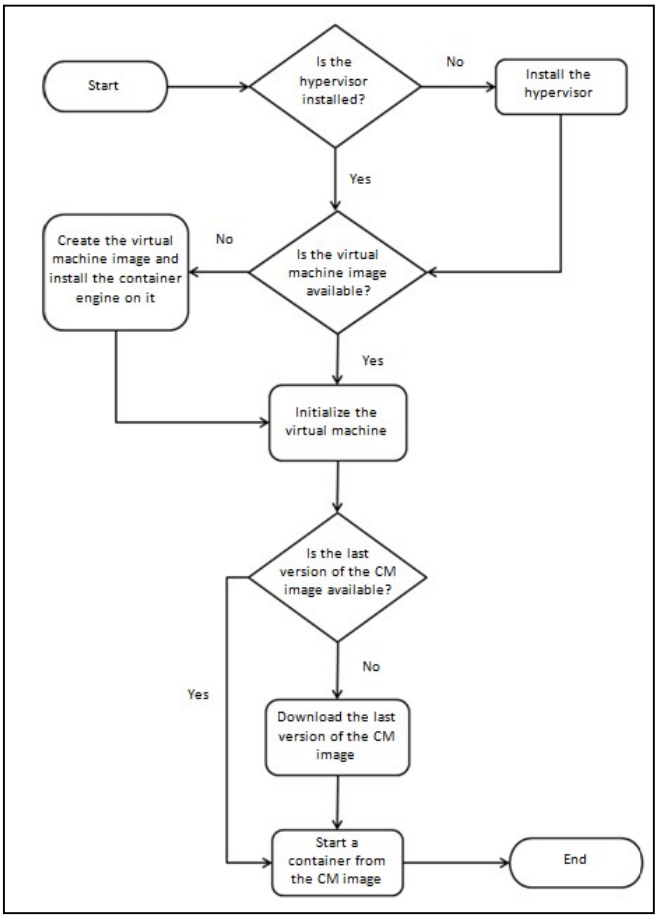


Figure 5. Steps performed by the VMM scripts.

VII. Case Study: The Prokaryotic Genomics and Comparative Genomics Analysis Pipeline (PGCGAP)

In this section, we describe our experience in recreating a computational environment called PGCGAP, the Prokaryotic Genomics and Comparative Genomics Analysis Pipeline, following the guide presented by H. Liu et al. in [44]. Also, we describe how we recreate the same environment following the guidelines defined by the ECF framework.

A. Material and Methods

We performed both implementations on three different physical machines. The machine one (M1) is a PC notebook configured with Microsoft Windows 10 Professional Edition 64-bit operating system, an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz processor, 16 GB of RAM and a hard disk 512 GB SSD. The machine two (M2) is a PC notebook configured with Microsoft Windows 10 Home Edition 64-bit operating system, an Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz processor, 16 GB of RAM and a hard disk 512 GB SSD. The last one, machine three (M3), is a PC notebook configured with Linux Fedora v34 64-bit operating system, an Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz processor, 16 GB of RAM and a hard disk 512 GB SSD.

In [44], the authors split the paper that presents the PGCGAP into two parts. The first part is related to the step-by-step defined by the protocol to provisioning the pipeline. The second part uses the pipeline provisioned to execute a bioinformatics application, processing a significant amount of data and generating results. As our focus is on providing the computational environment, we considered only the first part of the paper in our analysis and comparison.

The comparison of the two implementations considered time consumption, efforts, manual intervention, platform-agnosticism, and portability.

B. Implementing the Original PGCGAP

The PGCGAP is a protocol that guides researchers on implementing a scientific computational environment that supports applications related to genomics and comparative genomics analyses of microbes. This protocol comprises a set of genomic analysis software packages, scripts developed by the PGCGAP's creators, and a guide that specifies configurations and software installations that have to be performed by the researchers in a correct sequence to reproduce the environment. The PGCGAP is free and open-source software licensed under GPLv3. All the third-party software packages used to create the protocol are open source. The source code is available on Github [45].

In the paper, the authors demonstrate the protocol's applicability on a Linux Ubuntu 18.04 operating system. But, they used a machine configured with Microsoft Windows 10 and a feature called Windows Subsystem for Linux (WSL) [46] to create the virtual machine that supported the PGCGAP implementation. The Linux Ubuntu 18.04 OS was installed from the Microsoft Store.

As mentioned before, we reproduced the steps of the paper on three different physical machines. The M1 PC notebook is configured with Microsoft Windows 10 x64 OS, and it has the version 1 of the WSL installed. The M2 is configured with Microsoft Windows 10 x64 OS and WSL version 2. On both PCs were created virtual machines with Linux Ubuntu 18.04 OS installed from the Microsoft Store. Even though the authors did not specify if they used version 1 or 2 of the WSL, we decided to test the PGCGAP on both to verify if it can work adequately on any version. Regarding the M3 PC notebook, it is configured with Linux Fedora v34 x64 OS. We considered testing the protocol on the M3 machine to extend the research and check if it can work correctly on other Linux distributions that are not running on top of the Microsoft Windows Subsystem for Linux.

The PGCGAP protocol defines twelve steps that have to be executed by a researcher when provisioning the environment. These steps include installations of the WSL, the Linux OS, and third-party package software like Miniconda and the PGCGAP in itself. Also, it includes configurations that have to be included in files and performed in a command-line terminal. The authors reported that all the steps were performed in around sixty minutes.

On M1 PC, the provisioning of the PGCGAP environment failed. It presented a corruption error during the eleventh step that corresponds to the setup of the COG database.

All the steps to provision the PGCGAP environment were successful on M2 PC. It was necessary 80 minutes to perform the entire procedure. But, it is essential to highlight that it was needed 30 minutes more to install and configure version 2 of the WSL before executing the PGCGAP steps [47]. This was not an action foreseen by the protocol, but if we consider it, the whole process took 110 minutes in total. In terms of portability, we exported an image of the virtual machine created by the WSL to a zip file around 13.60 GB. Using this file, we could import the virtual machine on a fourth PC notebook (M4), used in this part of the experiment only for the portability test. It was configured with Microsoft Windows 10 operating system, WSL version 2, an Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz processor, 16 GB of RAM and a hard disk 512 GB SSD. After to import the virtual machine from the zip file, it was possible to initialize the PGCGAP environment successfully.

The provision of the PGCGAP on M3 PC was performed successfully, as well. To execute all the steps on this machine it was needed 81 minutes. This is a very close result that we obtained on M2 machine for this part of the provision. In this case, there was no extra installations and configurations to be considered. The portability of the PGCGAP from this machine is not possible, once the environment was provisioned directly on a physical machine.

The implementation of the PGCGAP protocol on the three machines mentioned before was performed manually, according to the steps presented in [44].

C. Implementing the PGCGAP Based on the ECF

For the development of the CM and VMM that will be described in this topic, we used the M4 PC notebook, already mentioned and detailed before.

As defined by the ECF framework, the first step on creating a computational environment is to develop the container module following the CM implementation guide.

For the analysis phase, it was necessary 30 minutes to review and fill the form with the requirements of the environment once most of them are described in [44]. Table III shows the form with the questions and answers used to develop the container module for the PGCGAP environment.

TABLE III. PGCGAP'S CM FORM

Question	Answer	Version
Which container engine will be used?	Docker and dependencies	20.10
Which will be the base image of the containers?	Ubuntu Linux	18.04
Which programming languages and compilers will be used?	Python and dependencies	3.7.6
Which libraries, packages and third part software have to be installed?	Miniconda and dependencies	3
Is it necessary to perform any configuration? In which files?	Add into .bashrc: export OMPI_MCA_opal_cuda_support=true	N/A
Is it necessary to copy any files into the container? Which files?	pgcgap_latest_env.yml	N/A

Based on the form shown in Table III, we could write the source code that defines the infrastructure needed to develop and run the scientific application. Practically, we developed the Dockerfile, a prerequisite necessary to generate the Docker container that will embed all the PGCGAP computational environment. Also, the Dockerfile works as part of the documentation. For programming and documenting the environment, it was necessary around 60 minutes.

The next step consisted of generating the container image based on the definitions of the Dockerfile. Docker performed this operation in 49 minutes, and the final base image file had a size of 8.14 GB. With the container's image ready to be used, it was necessary to execute a set of tests to ensure that a properly PGCGAP environment was being provisioned. We had to test the upload of the image to the Docker Hub, its download from the Docker Hub, and the instantiation of containers from the downloaded base image. Also, we ran some commands on the PGCGAP environment to verify that all the components were adequately installed. The image test operation was performed in 85 minutes.

Considering all the phases executed in the CM development, achieving a successful result took around 224 minutes.

With the container module working correctly, we started to work on the virtual machine module following the VMM guide. Initially, we analyzed the requirements needed to create a virtual machine capable of supporting the PGCGAP container, considering the bioinformatics profile of the applications that will run on this environment. Despite the ECF framework being tool-agnostic, it was designed for Linux containers. In this context, we opted to use only free and open-source software tools commonly used by the scientific community, as shown in Table IV. This step was performed in 30 minutes.

TABLE IV. PGCGAP'S VMM FORM

Question	Answer	Version
Which hypervisor will be used?	Virtualbox	6.1.32
How much memory will be allocated for the virtual machine?	8 GB	N/A
How much CPUs will be dedicated to the virtual machine?	2 CPUs	N/A
Which operating system will be installed on the virtual machine?	Ubuntu Linux	18.04
Which IaC tools will be used to automate the provisioning of the environment?	Vagrant and dependencies	2.2.19
	Ansible and dependencies	2.12.2
Other resources	Shell-scripts Linux (main script)	N/A
	Shell-scripts Windows (main script)	N/A

For this module, the first step was to implement a script used to start the PGPGAP environment, the main script. As our intention was to perform tests on Linux and MS-

Windows machines, we had to develop the main script for both operating systems. The script verifies if the PC has the Virtualbox installed. If not, the hypervisor is downloaded and installed on the machine. After, it verifies if Vagrant and Ansible are installed. In the negative case, the installation is performed, followed by the provisioning of the virtual machine configured with Ubuntu Linux. Otherwise, Vagrant only will start the virtual machine. In order to permit Vagrant to create the virtual machine, we had to specify the configurations and installations required in a file called Vagrantfile, as shown in Fig. 6. In this file, we defined the amount of RAM and the number of CPUs must be allocated for the VM. Also, we requested the installation of the Docker engine using Ansible. Having the VM running up, the script downloads the CM from the Docker Hub, if needed, and starts a container that embeds the PGCGAP environment.

```
##########################################################
# This Vagrantfile defines the configuration that will  #
# be used by the virtual machine module (VMM) to        #
# demonstrate the provision of the PGCGAP Environment   #
# through the implementation based on the               #
# Environment Code-First Framework                      #
##########################################################
Vagrant.configure("2") do |config|

  # Define the virtual machine image
  # Ubuntu Bionic 18.04 64 bits
  config.vm.box = "hashicorp/bionic64"

  # Define the hostname and the network
  config.vm.define :ecf do |ecf_config|
    ecf_config.vm.hostname = "ecf"
    ecf_config.vm.network :private_network,
                          :ip => "192.168.33.10"

    # Define the Ansible configuration
    ecf_config.vm.provision "ansible_local" do |ansible|
        ansible.playbook = "bio_ecf.yml"
        ansible.verbose = "vvv"
    end

    # Define the provider (virtualbox), quantity of memory,
    # and how many CPUs will be used on the VM
    config.vm.provider "virtualbox" do |domain|
        domain.memory = 8192
        domain.cpus = 2
    end
  end
end
```

Figure 6. Vagrantfile implemented for the virtul machine module.

The codification of all parts that compose the VMM, considering the scripts for Linux and Windows and the configuration files for Vagrant and Ansible, took around 280 minutes to be concluded. The scripts developed in this phase had to be tested individually and together, consuming around 350 minutes. This time includes the tests performed with the CM and the VMM together. Considering the time needed to implement both modules, CM and VMM, the total time was 884 minutes.

After the implementation of the CM and the VMM, we were ready to start the tests on the three PC notebooks described before: M1, M2 and M3. We started downloading the main script from the repository for both operating systems, MS-Windows and Linux. Actually, it is the only file that has to be downloaded manually by a researcher that wants to recreate the environment. The PGCGAP environment was provisioned automatically and succesfully by running this script on the three machines. Considering a scenario where all the software components (e.g., Virtualbox, Ansible, Vagrant, CM) had to be downloaded to make the environment available, this operation took 92 minutes on M1, 89 minutes on M2, and 95 minutes on M3. It is essential to highlight that these times can vary depending on the download capacity of the internet connection used to obtain the CM and the VMM. Both of them have to be downloaded, and, in this case, it was used an internet connection with 27 Mbps of download speed.

VIII. Discussion

First of all, it is essential to clarify that the ECF framework's primary goal is to enhance the reproducibility using the IaC approach. In this way, it focuses on guiding original researchers in providing a computational environment that is easily reproducible by them and other researchers. By them, when new members have to be integrated into a research team, for example. And by other researchers when they want to reproduce published results of a research papers. Of course, to enhance reproducibility and develop means that allow to recreate computational environments efficiently, a great effort from the researchers responsible for the provision in terms of learning and programming IaC tools will be necessary. The comparison presented in this topic can not be interpreted literally, but from two points of view, one from the original researcher that is creating the environment by programmatic ways, and another from the researcher that is reproducing it. The work presented in [44], only shows the second perspective.

In terms of time consumption, from the point of view of the original researcher, it was necessary around fifteen hours to build the entire computational environment that supports the PGCGAP usinf the ECF framework. It is essential to highlight that it was designed to assist those directly involved in research development and anyone who wants only to run an application and verify published results. From this second point of view, the effort necessary would be simply downloading and executing only one script to get the computational environment ready to use. Our practical test on this operation consumed 92 minutes, on average, considering the three machines used in the tests (M1, M2, and M3). By following the method presented in [44], our provisioning of the environment took on average 80.5 minutes, remembering that we did not have success in implementing it on the M1 PC notebook. The authors reported in [44] a total time of 94 minutes to provisioning it.

Another concern of the ECF framework is reducing the manual intervention when provisioning computational environments. Our experience in reproducing the original PGCGAP environment proved that, by following this method, all the steps must be performed manually. From enabling the WSL resource on MS-Windows operating system, installing the Ubuntu operating system, downloading and installing the Miniconda platform, adding configuration on some specific Linux files until the installation of the PGCGAP, all of them were performed in a manual way. And, this is not a good practice when trying to produce reproducible environments. On the contrary, manual intervention is one of the leading causes of issues like the

snowflake servers and snowflake systems mentioned earlier. Implementing the PGCGAP based on the guidelines of the ECF framework showed us that it is possible to provide an entire computational environment automatically and programmatically, reducing the manual intervention to a download and execution of only one script.

The platform-agnosticism is another relevant topic that is covered by the ECF framework. The framework focuses mainly on the three most used operating systems by scientific researchers: Linux, MS-Windows and MacOS. But, it does not exclude other platforms like HPC and cloud computing. Once the framework defines a standard layer represented by a virtual machine, the only condition to run an ECF environment is to support virtualization. This abstraction turns different platforms in one common platform using the same distribution of the Linux operating system. The Linux container representing the computational environment can be instantiated from this point, permitting that the applications consistently produce the same results. The PGCGAP environment implemented by the ECF framework could be provisioned successfully on the three machines used in our experiments. Differently, the original PGCGAP could be provisioned only on two of the three machines due to an issue related to version 1 of the WSL resource, which is part of the MS-Windows and is a vital tool of this proposal.

In terms of portability, as the ECF framework was designed to be tool-agnostic, one implementation can be more portable than another depending on the way they were implemented. For example, using Python instead of operating system scripts to develop the VMM produces a more portable environment. In our case, we decided to use shell scripts to implement the main script of the VMM due to our high knowledge of this subject. Choosing another programming language like Python would require more time to learn, implement and test this deliverable. In this way, we had to create one main script for Linux and another for the MS-Windows platform because the hypervisor (Virtualbox) installation is different on both operating systems. All the source code produced is common for any platform from this point ahead. The configuration files created for Vagrant, Ansible, and Docker, for example, will be the same on provisioning the environment for any operating system. The main script is the only deliverable that needs to be download and executed manually by a researcher that wants to reacrete the environment. When it is executed, all the softwares that compose the infrastructure (e.g., Virtualbox, Vagrant, and Ansible) are downloaded and installed automatically from official repositories. The CM is downloaded from the Docker Hub. That means the environment is always provisioned with the same components and versions obtained from the same sources. It can be installed on physical machines or infrastructures supported by cloud computing. Based on what we produced in our experiment, it is limited to Linux and MS-Windows. But, we can extend this coverage only by creating the main script for other platforms (e.g., MacOS, Solaris). Once the original researchers provided access to the source code of the environment, it is possible yet, for other researchers to reproduce it by compiling this code. This practice is another advantage of the ECF framework. There is no need to perform the portability manually, using traditional means like copying a file from one computer to another. On the other hand, this is the only way presented by the original PGCGAP, considering that both machines have MS-Windows 10 operating system and WSL version 2 installed. In this case, it is important to remember that, in our experiment, WSL generated a file with 13.60 of size in the export operation. The container image file that represents the CM had 8.14 GB. A difference of 40.14% between them.

The documentation is another positive point of using the ECF framework to develop the PGCGAP environment. As the framework uses the IaC approach, all the components of the environment are based on code. In this way, we described and explained the environment, installations, and configurations, into the source-code files we had created for Vagrant, Docker, Ansible, and the shell scripts. The documentation is fundamental to guide those who need to recreate the environment, but also for anyone that wants to understand how the environment was developed.

Of course, we can not show only the benefits of the ECF framework. There are costs on adopting it. First of all, it is important to highlight the time that the original researchers must dedicate to programming the components of the environment. As mentioned earlier, it took about fifteen hours to develop and test the source code, given our high level of expertise in the programming languages and tools used in the experiment and the Infrastructure-as-Code approach. The development and test phases tend to be higher when researchers are not information technology specialists (e.g., biologists, chemists, and archeologists). This implies the factor that we consider to be the second higher cost: the time and effort needed to learn about the programming languages, IaC tools and software engineering practices.

IX. CONCLUSION

In [44], the authors presented a step-by-step on recreating an entire computational environment that supports scientific research. That is what we expect from an executable paper in terms of documentation. However, it is based on a manual intervention approach which is not good reproducibility practice. When recreating an environment, increasing reproducibility implies substituting the actions manually performed by automatic means.

The framework proposed in this paper has as primary intention to help researchers enhance the reproducibility of their work regarding the provision of the computational environment. Our study contributes by mitigating typical issues such as the absence of documentation, platform dependency, and manual intervention. The central idea is provisioning the environment based on the Infrastructure-as-Code approach instead of using traditional means. The pre-defined homogeneous architecture permits us to have a big picture of the environment. And the practical guidelines conduct researchers on developing environments that are more reproducible, isolated, portable, and independent of any software and hardware platform.

Our goal in proposing the ECF framework is to support the development of new research works and help researchers

migrate papers already published based on traditional provisioning approaches to this new way easily reproducible.

In the case study presented, the ECF framework proved in practice that it is possible to provide an entire computational environment programmatically with minimal intervention of the researcher who wants to recreate it. In the results, we can testify its benefits, especially regarding the behavior of the environment that was the same on the three machines used in our experiment. This kind of evidence allows us to understand how valuable it is to mitigate manual intervention because we will always have the same environment many times as we reproduce it.

The adoption of the ECF framework has costs. Firstly, we have to highlight the time and effort from the scientist-developer side when producing and testing the code that will generate the environment. These activities require them to dedicate a lot of time and attention. Besides, it is necessary a continuously learn behavior from the scientist-developer, which we consider the second cost when adopting the framework. Developing competence in software engineering practices, programming languages, and IaC tools is challenging and time-consuming. But, it is also a gain for researchers once they are preparing themselves for a new era based on open science principles.

We recommend using open source software for researchers who desire to implement their computational environments following the ECF guidelines. Besides being aligned with the open science principles, open source tools in the general count with large communities supporting the users, accelerating the learning process, and helping them when they face technical problems. The use of mature tools already approved by the scientific community (e.g., Docker, Virtualbox, and Python) is also recommended. Compared with more recent tools, they tend to have fewer technical issues, and there is more documentation and forums to guide new users.

We consider a significant contribution of the ECF framework the educational role that it can have in helping to prepare future generations of researchers on creating more transparent and reproducible research that aggregates value and benefits the scientific community.

As future work, we propose implementing computational environments for different domains of science to help to improve the ECF framework.

REFERENCES

[1] D. A. Gomes, P. Mestre, and C. Serôdio, “Infrastructure-as-Code for Scientific Computing Environments,” CENTRIC 2019: The Twelfth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, Nov. 2019.

[2] J. A. Papin, F. Mac Gabhann, H. M. Sauro, D. Nickerson, A. Rampadarath, “Improving reproducibility in computational biology research,” PLoS Comput Biol 16(5): e1007881, 2020, https://doi.org/10.1371/journal.pcbi.1007881.

[3] B. A. Nosek et al., “Promoting an Open Research Culture,” Science, New York, N.Y., 348, 2015.

[4] L. P. Freedman, I. M. Cockburn, T. S. Simcoe, “The Economics of Reproducibility in Preclinical Research,” PLOS Biology 13(6): e1002165, 2015, https://doi.org/10.1371/journal.pbio.1002165.

[5] European Commission. Reproducibility of scientifc results in the EU. [Online] Available from: https://op.europa.eu/en/publication-detail/-/publication/6bc538ad-344f11eb-b27b-01aa75ed71a1. [Accessed: 2022.05.31].

[6] J. R. F. Cacho and K. Taghva, “The State of Reproducible Research,” in Computer Science, 17th International Conference on Information Technology – New Generations (ITNG 2020), Advances in Intelligent Systems and Computing, Volume 1134, Springer, 2020.

[7] M. Munafò et al., “A manifesto for reproducible science,” Nat Hum Behav 1, 0021, 2017.

[8] J. C. Burgelman et al., “Open Science, Open Data, and Open Scholarship: European Policies to Make Science Fit for the Twenty-First Century,” Frontiers in Big Data, Volume 2, 2019.

[9] J. R. F. Cacho and K. Taghva, “Reproducible research in document analysis and recognition,” in Information Technology-New Generations, Springer, Berlin, pp. 389–395, 2018.

[10] M. Baker, “1500 scientists lift the lid on reproducibility,” Nature News 533 (7604), 452, 2016.

[11] C. Boettiger, “An introduction to Docker for reproducible research, with examples from the R environment,” ACM SIGOPS Oper. Syst. Rev., 49, 2014.

[12] S. M. Powers and S. E. Hampton, “Open science, reproducibility, and transparency in ecology,” Ecological Applications 29(1):e01822, 2019.

[13] D. L. Donoho, “An invitation to reproducible computational research,” Biostatistics (Oxford, England), 11, 2010, pp. 385-8.

[14] A. Brinckman et al., “Computing environments for reproducibility: Capturing the “Whole Tale”,” Future Generation Computer Systems, Volume 94, pp. 854-867, 2019.

[15] K. Morris, “Infrastructure as Code: Managing Servers in the Cloud,” 1st ed. O’Reilly Media, Inc., 2016.

[16] C. Collberg, T. Proebsting, G. Moraila, A. Shankaran, Z. Shi, and A. M. Warren, “Measuring Reproducibility in Computer Systems Research,” Department of Computer Science, University of Arizona, Technical Report. [Online]. Available from: http://reproducibility.cs.arizona.edu/tr.pdf [Accessed: 2022.05.31]

[17] T. Glatard, L. B. Lewis, R. Ferreira da Silva, R. Adalat, N. Beck, C. Lepage, and A. C. Evans, “Reproducibility of neuroimaging analyses across operating systems,” Frontiers in Neuroinformatics, 9, 12, 2015, doi:10.3389/fninf.2015.00012.

[18] C. Riccomini, D. Ryaboy. “The Missing README: A Guide for the New Software Engineer,” 1st ed. No Starch Press, 2021.

[19] D. C. Ince, L. Hatton, and J. Graham-Cumming, “The case for open computer programs,” Nature. 2012; 482(7386):485-488. Published 2012 Feb 22.

[20] B. Marwick, “Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation,” J Archaeol Method Theory 24, 424–450, 2017, https://doi.org/10.1007/s10816-015-9272-9.

[21] J. Segal and C. Morris, "Developing Scientific Software" in IEEE Software, vol. 25, no. 04, pp. 18-20, 2008, doi:10.1109/MS.2008.85.

[22] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl and G. Wilson, "How do scientists develop and use scientific software?," 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009, pp. 1-8, doi:10.1109/SECSE.2009.5069155.

[23] L. Nguyen-Hoan, S. Flint, and R. Sankaranarayana, “A survey of scientific software development,” In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10). Association for Computing Machinery, New York, NY, USA, Article 12, 1–10, 2010, doi:https://doi.org/10.1145/1852786.1852802.

[24] Z. Merali, “Computational science: Error, why scientific programming does not compute,” Nature 467, 7317, 2010, https://doi.org/10.1038/467775a.

[25] E. M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, J. C. Carver, "Software engineering practices for scientific software development: A systematic mapping study," Journal of Systems and Software, Volume 172, 2021, 110848, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2020.110848.

[26] Á. L. García and E. F. del Castillo, "Analysis of scientific cloud computing requirements," in Proceedings of the 7th Iberian Grid Infrastructure Conference, Madrid, Spain, Sep. 2013, pp. 147-158.

[27] B. S. Cole and J. H. Moore, "Eleven quick tips for architecting biomedical informatics workflows with cloud computing," PLOS Computational Biology, vol. 14, issue 3, Mar. 2018.

[28] D. Clark, A. Culich, B. Hamlin, and R. Lovett, "BCE: Berkeley's Common Scientific Compute Environment for Research and Education," in Proceedings of the 13th Python in Science Conference, Austin, USA, Jul. 2014, pp. 5–12.

[29] B. Howe, "Virtual Appliances, Cloud Computing, and Reproducible Research," Computing in Science & Engineering, vol. 14, no. 4, pp. 36-41, Jul.-Aug. 2012.

[30] J. R. Fonseca and K. Taghva. "Reproducible Research in Document Analysis and Recognition," Advances in Intelligent Systems and Computing: Information Technology - New Generations, Springer, 738 389-395, 2018

[31] Amazon Web Services. Introduction to DevOps on AWS, White Paper, 2014. [Online]. Available from: https://d1.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf [Accessed: 2022.05.31]

[32] Amazon Web Services. Infrastructure as Code, White Paper, 2017. [Online] Available from: https://d1.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf [Accessed: 2022.05.31]

[33] S. Nelson-Smith, "Test-Driven Infrastructure with Chef," 2nd ed. O'Reilly Media, Inc., 2013.

[34] K. Morris, "Infrastructure as Code: Dynamic Systems for the Cloud Age," 2nd ed. O'Reilly Media, Inc., 2020.

[35] IBM Cloud Education. Infrastructure as Code, Blog, 2019. [Online] Available from: https://www.ibm.com/cloud/learn/infrastructure-as-code [Accessed: 2022.05.31]

[36] IBM Cloud Education. Containers vs. Virtual Machines (VMs): What's the Difference?, Blog, 2021. [Online] Available from: https://www.ibm.com/cloud/blog/containers-vs-vms [Accessed: 2022.05.31]

[37] Red Hat Topics Website. Containers vs VMs, 2020. [Online] Available from: https://www.redhat.com/en/topics/containers/containers-vs-vms [Accessed: 2022.05.31]

[38] Microsoft Technical Documentation Website. Containers vs. virtual machines, 2021. [Online] Available from: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm [Accessed: 2022.05.31]

[39] K. Wiebels and D. Moreau, "Leveraging Containers for Reproducible Psychological Research." Advances in Methods and Practices in Psychological Science, Apr. 2021, doi:10.1177/25152459211017853.

[40] X. Qiao, Z. Li, F. Zhang, D. P. Ames, M. Chen, E. J. Nelson and R. Khattar, "A container-based approach for sharing environmental models as web services," International Journal of Digital Earth, 14:8, 2021, 1067-1086, doi:10.1080/17538947.2021.1925758.

[41] Red Hat Topics Website. What's a Linux container?, 2018. [Online] Available from: https://www.redhat.com/en/topics/containers/whats-a-linux-container [Accessed: 2022.05.31]

[42] J. Sparks, "Enabling Docker for HPC," Concurrency and Computation: Practice and Experience, Dec. 2018, https://doi.org/10.1002/cpe.5018.

[43] Docker Documentation Website. Docker Desktop overview. [Online] Available from: https://docs.docker.com/desktop/ [Accessed: 2022.05.31]

[44] H. Liu et al., "Build a bioinformatics analysis platform and apply it to routine analysis of microbial genomics and comparative genomics," Protocol Exchange, 2020, https://doi.org/10.21203/rs.2.21224/v5.

[45] Github of the Project PGCGAP. [Online] Available from: https://github.com/liaochenlanruo/pgcgap [Accessed: 2022.05.31]

[46] Official documentation of the Windows Subsystem Linux. [Online] Available from: https://docs.microsoft.com/en-us/windows/wsl/ [Accessed: 2022.05.31]

[47] Windows Subsystem Linux's documentation on how to migrate from version 1 to version 2. [Online] Available from: https://docs.microsoft.com/en-us/windows/wsl/install-manual/ [Accessed: 2022.05.31]

Hybrid Transactional and Analytical Processing Databases - State of Research and Production Usage

Daniel Hieber
Dept. of Computer Science
Aalen University
Aalen, Germany
Email: daniel.hieber@studmail.htw-aalen.de

Gregor Grambow
Dept. of Computer Science
Aalen University
Aalen, Germany
Email: gregor.grambow@hs-aalen.de

***Abstract*—The combination of Online Transactional Processing and Online Analytical Processing into one system is an emerging area in database research called Hybrid Transactional and Analytical Processing databases (HTAP, OLxP). Both Gartner and Forrester Research see disruptive potential in this technology as it provides important advantages. These include the elimination of redundant data sets for analytical and live data as well as the reduction of the total cost of ownership of analytics systems. The development of HTAP databases resulted in various advances in the database sector like the creation of new index and data structures or improvements of existing concurrency control implementations. However, there is a great variety regarding many architectural aspects in different HTAP systems. Examples include implementations of concurrency control, query handling, or scaling paradigms ranging from scaled-up single server systems using Multi Version Concurrency Control to scaled-out cluster based systems using last writer wins approaches. This contribution provides a general overview of contemporary HTAP implementations. On the one hand, different fundamental technical aspects are presented and compared in detail. On the other, it goes beyond a standard literature review by also presenting an overview of production ready HTAP systems, including both free and commercial systems.**

Keywords*—*Hybrid Transactional Analytical Processing; HTAP; Database; Literature Study; OLxP.

I. Introduction

The need to analyse data in realtime and not to rely on copies of old databases combined with the growing wish of companies to gather all data in one database lead to the rise of Hybrid Transactional Analytical Processing (HTAP) Databases. In our initial systematic literature review from 2020 on this topic, we provided a comprehensive summary focused on the research of these systems [1]. While HTAP as a term was coined by Gartner [2] in 2014 and even before that there had already been active research in the area these databases are still heavily evolving and only starting to get a foothold in production systems. Therefore, in this work, we extend our previous literature review with the current state of HTAP, highlighting new research conducted, but also introducing an overview of currently available HTAP systems for use in production use cases, including both free and commercial systems.

Solving the problems of keeping data in two separated databases and at the same time reducing the total cost of ownership by introducing one unified system instead, HTAP efficiently combines Online Transactional Processing and Online Analytical Processing capabilities in one system. Therefore, both Gartner [3] and Forrester Research [4] see disruptive potential in HTAP. A trend already projected to the industry, e.g., with commercial solutions provided by two of the world leaders SAPs HANA database [5] and Tableaus HyPer [6] integration.

In this work the basics of the different fundamental architectures for HTAP database systems like HyPer [7] and SAP HANA [8] are explained and different approaches regarding the concrete implementations as well as optimization approaches are introduced in form of a refined version of our earlier systematic literature review. Further we provide insight into the production implementation of HTAP databases like SAPs HANA [5] and Tableaus HyPer [6] systems. The aim of this paper is to reflect the current state of research in an ordered way, as well as to highlight important decisions leading to todays implementations and their production use.

This remainder of this paper is organized as follows: Section 2 provides background on database processing paradigms covered in this paper. Section 3 describes the underlying literature review process in detail. Section 4 discusses the findings and provides a overview of the current development and research state of HTAP.

The section is further separated into subsection ordering findings by the area of HTAP it deals with:

- IV-A Fundamental Architecture (containing scaling paradigms, data/table structures and ways of saving/partitioning data)
- IV-B Concurrency
- IV-C Garbage Collection
- IV-D Query Handling (containing query languages, general optimization approaches, query processing in differently scaling systems)
- IV-E Indices
- IV-F Big Data on HTAP
- IV-G Recovery, Error Handling and Logging
- IV-H Benchmarking of HTAP Systems
- IV-I Stream Processing with HTAP Systems
- IV-J HTAP as a Service
- IV-K Future trends of HTAP development
- IV-L Open Source and Free Versions

Section 5 then provides insights on the current sate of HTAP databases regarding their production usage.

Finally, Section 6 summarizes the provided work, supplying all required information in a short form.

II. Background

This section provides some background information regarding the database processing paradigms covered in this paper.

A. Online Transaction Processing

Online Transaction Processing (OLTP) describes a category of data processing that is focused on transaction-oriented tasks. The workload is heavily write oriented, consisting of insert, update and delete operations. The size of data involved is usually relatively small, while the amount of transactions can be massive.

Features like normalization and ACID are required by OLTP to function efficiently. Besides fast processing and highest availability, data consistency is also one of the most important features of OLTP databases.

B. Online Analytical Processing

Online Analytical Processing (OLAP) is focused on complex queries for dataset analysis. The workload is read heavy and can include enormous datasets. In order to efficiently analyse such big amounts of data, intelligent indexing and fast read times are necessary. OLAP workloads are resource heavy and require high performance systems.

C. Hybrid Transactional Analytical Processing

Hybrid Transactional Analytical Processing (HTAP) combines both OLTP and OLAP in one database. Therefore, writing and analyzing data is efficiently handled in the same database, removing the need to run two separate systems and thereby reducing implementation efforts, maintenance and cost. However, the resource intensive workload of OLAP queries and the required high availability of OLTP compete with each other and require new solutions to work on the same system.

III. Literature Review Methodology

While the term HTAP was first used 8 years ago in 2014, many researchers still did not adopt it and use other phrases like OLxP, HOAP or OLAP and OLTP hybrid databases. To ensure a comprehensive and high-quality literature base for the review, several searches were carried out with different search therms.

In the study, Kitchenham's systematic review procedure [9] was employed. The following steps were pursued:

1) Determining the topic of the research
2) Extraction of the studies from literature considering exclusion and inclusion criteria
3) Evaluation of the quality of the studies
4) Analysis of the data
5) Report of the results

As a topic the current state of HTAP databases was selected, summarizing the research conducted on the topic. To determine the best suited source and search query for the data search of the literature review multiple data sources (including Google Scholar, Semantic Scholar and IEEE) were tested with a multitude search queries.

The reviewing process (Figure 1) was conducted via Google Scholar as this search engine provided the highest quantity and quality of research for the topic. Further the Google Scholar searches included most of the results the other data sources contained. Searches with other search engines and data sources where either lacking a sufficient quantity or quality to conduct a meaningful literature review.

To counter the aforementioned issues regarding the insufficient usage of the therm Hybrid Transaction Analytical Processing databases in the research the literature acquisition was split into three queries using different search structure and therms. While very similar the returned research of each search query was mostly disjunct.

While this approach increased the number of relevant papers found it does not provide all-encompassing literature findings. This is mainly due to skipped keywords in the research papers themselves. E.g., Umbra [10] only describes itself as a further development of HyPer [7] but never mentions HTAP itself in the paper, while its extension [11] actively mentions HTAP. Therefore, the extension was found, while the main paper is not included in this study.

In order to prevent the absence of relevant systems not found by the systematic literature search itself, the sections "Open Source and Free Versions" as well as "Production Ready HTAP Databases" also contain HTAP systems not found by the systematic literature search if they provided relevant free/open source solutions or are used in production systems.

The search was carried out using (1) "htap" "data warehouse" OR "OLTP" "OLAP" (returning 183 entries), (2) HTAP OR OLAP OLTP hybrid database (returning 200 entries) and (3) hybrid transactional analytical processing (returning 200 entries).

In this first search only publications from 2010 to August 2020 were considered. Queries 2 and 3 returned more papers, but were reduced to the 200 most recommended papers, since quality and relevance were continuously decreasing. To provide an up-to-date overview of the current research in this paper slightly refined versions of the search queries were executed again for the time span from 2020 to October 2021 (cf. Figure 1). To provide a comprehensible and reliable literature review only publicly accessible papers or papers available with general institutional access were taken into account. The conducted papers were further reduced to papers using the German or English language. These exclusion criteria left 147 (1), 178 (2) and 179 (3) papers from the first search as well as 70 (1), 7 (2) and 37 (3) papers from the second search to refine further. In this step, all papers already included in the first search were also removed from the findings of the second search.

Following a title and abstract based elimination was conducted. This pruned papers lacking a combination of required key words or only mentioning HTAP as a side note. After this elimination step, 55 (1), 44 (2) and 56 (3) papers (first search)

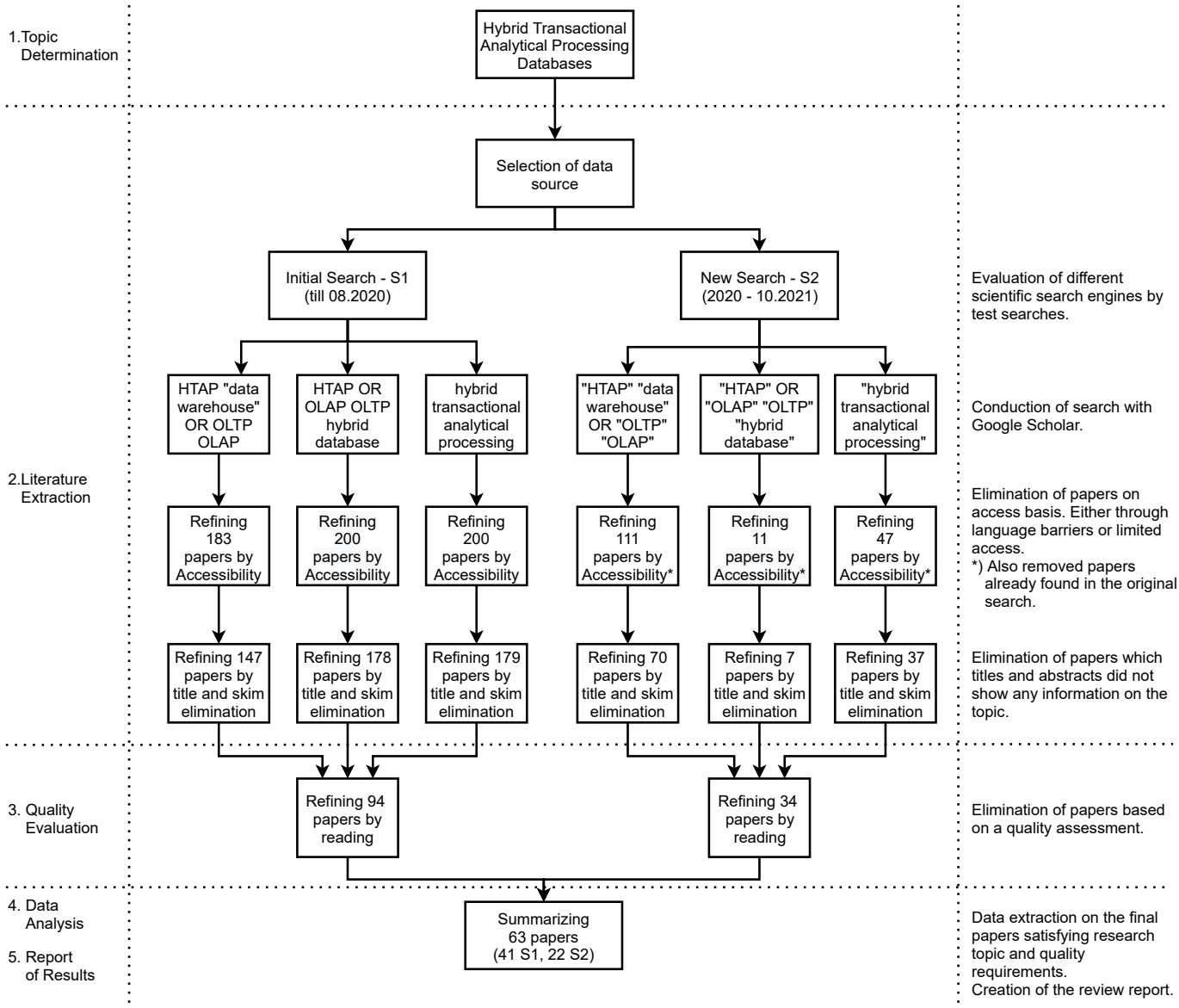


Figure 1. Literature Review Process.

as well as 30 (1), 4 (2) and 20 (3) papers (second search) were left for further analysis. Removing any duplications of papers found by two or more queries this left a total of 128 papers combined for a final review.

Of these 128 papers, 16 were found to be of insufficient quality (lacking evaluations or proper research procedures), and 30 did contrary to their title and abstract not focus on the topic of HTAP databases or on fundamental technologies for those. The 84 papers, which were found scientifically significant and fulfilling the quality requirements were finally reduced to 63, deducting papers providing only outdated non-fundamental information.

IV. Findings and Discussion

The methods to create HTAP databases, their functionality and their optimizations take many different approaches. The contents of the papers were organized into the following sections according to the kind of information provided.

A. Fundamental Architecture

HTAP databases build up a new database sector and there are many databases, which were newly developed for this workload, e.g., [7][12][13]. However, some existing databases also have been upgraded to handle HTAP workloads like SAP HANA [8], initially an OLAP database, and PostgreSQL [14] (as well as multiple systems building on it, cf. [15]), initially an OLTP database, proving that existing databases can be extended to handle HTAP.

Comparing the reviewed database architectures, two main storage paradigmas can be clearly identified with the reviewed solutions: (1) heavily main memory focused databases, keeping all of their (hot) data in memory like HANA [5], HyPer

[7], BatchDB [13] and Hyrise [16], as well as (2) cloud/shared disk data stores, keeping some data in memory but relying on a persistent out of memory data store accessible by all instances, e.g., Wildfire [17], TiDB [18] and Janus [19].

Further, a Non-Uniform Memory Access (NUMA) architecture is a base requirement for most main memory HTAP databases like SAP HANA [5], AIM [20], BatchDB [13], Hyrise [16] and HyPer [7] enabling multiple cores to access each others memory.

1) Scaling out and up: Another big difference in HTAP databases is their scaling approach. Systems like HyPer [7] (commercialized by Tableau), Poseidon [21] or Hyrise [16] are deployed on single servers utilizing NUMA to scale-up onto multiple cores, thus creating multiple nodes. This approach can reduce processing time as no data transfer between different servers is required and all data can be accessed in memory. As a downside however, large systems require a strong server with a large main memory. Both HyPer and Hyrise also provide scale-out approaches, normally keeping their OLTP processing on the main server, e.g., ScyPer [22].

The main memory database Polynesia [23] also follows a single server scale up approach. This happens by separating responsibility of workloads to different OLTP and OLAP "islands". By adding more islands the analytical and transactional throughput can be increased.

Like Hyrise and HyPer - SAP HANA [24] keeps the OLTP workload on one machine, utilizing NUMA to use as many cores as required and available, but implements scaling the OLTP workload out to other servers as a base feature. Using HANA Asynchronous Parallel Table Replication (ATR) the database distributes its data amongst multiple replicas enabling a more efficient OLAP approach.

BatchDB [13] also handles the OLTP workload on the main server. The OLAP workload can be either executed on a different node of the same machine, or an entirely different server.

Contrarily, Wildfire [25][26] (while initially commercialized as IBM DB2 Event Store not all Wildfire decisions seem to apply to Db2 Event Store anymore) utilizes a fully distributed approach. Heavily relying on Apache Spark and Apache Zookeeper, all requests pass Sparks API and get distributed across multiple Spark executors. These executors delegate the transactional and analytical requests to the Wildfire engine daemons. All daemons use their main memory as well as SSDs and are connected to one shared data storage, e.g., a cloud data store. With this approach more throughput can be achieved, but ACID on the other hand is no longer possible. One of the more recent researches on the Wildfire system, Wildfire-Serializable (WiSer) [27] also offers high availability besides HTAP. It is furthermore optimized for IoT workloads.

Like Wildfire, SnappyData [28] also uses Spark as a core component to scale out the system to a database cluster. Therefore, the system enables more information to be kept in memory without the need for one expensive server.

Janus [19] also uses a distributed setup but implements the query distribution on its own with execution servers. These delegate the query to a corresponding row partitioned server for OLTP workloads or a column partitioned server for OLAP workloads.

TiDB uses a refined multi Raft-Group-approach. The database is separated into multiple regions, each having their own Raft group. OLTP workloads are send to the group leader, which replicates them asynchronously to other OLTP followers and synchronously to an OLAP column store. Multiple Raft groups can share the same OLAP optimized column store. OLAP queries can then be executed using both, the OLTP leader and followers, as well as the OLAP database [18]. The primary scaling is achieved by splitting into more regions with their own Raft groups. Each group, however, only has one leader ingesting OLTP requests. OLAP requests can be handled by the whole Raft group as well as a read optimized column store. OLTP scaling is therefore only possible by splitting groups and creating more leaders.

Another unique approach is taken by AnyDB [29]. Instead of committing to one scheme they use servers with stateless nodes called AnyComponents (AC). These components can then take any required role (worker handling queries, query optimizer creating these queries). By adding more ACs to a server or by adding more servers they provide great scalability. The exchange between the ACs is handled by event streams buffered in queues.

VEGITO uses a shard-approach, distributing sets of a primary and backup OLTP stores with an OLAP store over multiple shared-nothing machines. The primary, backup and analytical store for the same key range however do not have to be situated on the same machine.

The PostgreSQL based Greenplum builds a cluster consisting of multiple PostgreSQL databases called segments [15]. One segment takes the role of the coordinator while the other segments build the actual database. Scaling is possible by simply adding more segments to the cluster.

2) Data/Table Structure: When dealing with OLTP and OLAP workloads, finding the right table format can be difficult. HTAP databases therefore employ different table and data structures. Wildfire [25] exclusively uses column oriented tables since they are the most efficient solution for OLAP workload. Db2 Event Store further heavily utilizes Apache Parquet and its encryption implementation [26].

SAP HANA [5] implements a row-store query engine and a column-storage engine to combine the advantages of both technologies. Thus, it is possible to save data in row or column tables. The column layout is the default, more optimized, option.

HyPer [30] and Hyrise [16] both use columnar stores with self implemented data models. Hyrise further presented a hybrid column layout in an older version [31], combining simple one-attribute-columns with rows. This is planned to be implemented again in the new version, but has low priority and is work in progress.

Opposed to this, PostgreSQL [14] continues to use its row data storage for OLTP, but has a column store extension for OLAP workloads, merging the delta from the row store

continuously in the column store. This is handled similarly by VEGITO [32] and TiDB [18]. However, column and row stores are situated on different servers for maximized query performance and safety (required by TiDB and possible by VEGITO).

While Greenplum also uses the default PostgreSQL row storage for fresh data, older data is moved to a column-oriented store and even older data can be stored in external storage systems like Hadoop [15].

Polynesia uses a row and column approach, where transactions are saved in a N-ary storage model while analytical data is saved in a three stacked memory approach [23]. The column data is stored in so called vault groups (consisting of four vaults each). Each column is then spread evenly across the vaults of a group. Following a decomposition storage model each vault further redundantly contains a dictionary to reduce lookup cost.

SnappyData [28] follows a hybrid approach, where the fresh data is stored in an in-memory row-store and is moved in an on-disk column-store after aging.

The Cloud data store Janus [19] is fully hybrid, utilizing row partitions for OLTP and column partition for OLAP. Via a redo-log inspired batching approach and a graph-based dependency management, the delta from the row replicas can be merged into the column replicas.

The Casper prototype [33] uses a tailored column layout to support mixed read/write workloads more efficiently. With this approach, runtime column adaptations are possible.

Flexible Storage Model (FSM) [34] presented a tile based architecture to allow a transition from OLTP optimized tables to OLAP optimized tables depending on the hotness of data. The data is saved in a row oriented manner at the beginning and, depending on the hotness, is tile-wise transitioned to an OLAP column oriented tile structure.

3) Saving and Partitioning Data: For scale-up focused databases, removing data from main memory to larger, more cost efficient stores (e.g., hard drives), or efficiently compressing its size, is crucial. HyPer uses horizontal partitioning and saves its hot data uncompressed on the main memory. The cold data can also be kept in memory. Instead of evicting data to a disk, the data is compressed into self implemented Data Blocks [30] and kept in main memory. However, it is possible to evict them to secondary storage solutions if preferred (e.g., non-volatile random-access memory) and use them as persistent backups. The compression technique is chosen based on the data actually saved in the Data Block.

Utilizing Small Materialized Aggregates (SMAs) including meta data like min and max values, irrelevant compressed data can easily be skipped in searches. If data cannot be skipped on SMA basis, Positional SMAs (PSMAs), another lightweight indexing structure developed by the HyPer team, can be used. These help to determine the range of positions in the Data Block were the relevant values are located.

Hyrise [31] solves this problem using horizontal partitioning and by saving data in 2 kinds of columns: Memory-Resident Columns for hot data in memory, allowing fast access, and uncompressed row-oriented Secondary Storage Column Groups for cold data on hard drives. As the cold data is saved uncompressed, the cost of accessing it is reduced in comparison to classical compressed approaches.

Furthermore, the data is organized in so-called chunks [16] similar to Data Blocks. Chunks can be mutable as long as they are not full. As soon as they reach their capacity they transition to an immutable append-only container. They also have indexes and filters on a per chunk basis like Data Blocks, allowing faster search and access operations.

In Poseidon, data is saved in non-volatile memory (NVM) as well as main memory [21]. New transactions are kept solely in main memory until the commit is conducted, then the final data is saved on NVM. With this approach the low latency of main memory can be used for all steps of an transaction, while at the same time the final data object is saved in a persistent way. Another project exploring NVM in the form of directly-attached-NVMe-arrays (DANA) was conducted by Haas et al. [35], optimizing LeanStore for DANA usage achieving promising results. However, updates of traditional disk-based systems seem to be no feasible option due to high CPU load.

Smart Larger Than Memory [36] stores cold data in files on the hard drive decoupled from the database. Modifications to the data are no longer possible. The entries can only be deleted. This happens via removing the reference entry in memory without accessing the cold data and thereby saving time. Updating cold data is possible, but the update is a hidden delete of the cold data index and an insert of new hot data. To fully take advantage of SmartLTM the read operations always check the main memory entries first. If the data cannot be found, cuckoo filters or SMAs are used to locate the data in the files on the hard drive.

In VEGITO data is stored in paged blocks called epochs [32]. The epoch counter is continuously increased (around every 15ms). Witch each new epoch the data from the old epoch either gets copied to a new page, if a change occurred in the block, or is simply referenced if no change occurred. This saves copy time as well as memory.

TiDB first saves deltas in memory. When the amount of small deltas increases they are merged to bigger deltas and moved to on-disk storage [18]. Like in F1-Lightning, data partitions can further be adjusted on the fly, splitting to large partitions or merging smaller ones for optimized performance.

Like most systems Db2 Event Store utilizes an approach were the most recent data is moved to the fastest storage and is moved down the chain (in total three zones) as it ages [26]. In the first step fresh data is saved on NVMe SSDs, then it is moved to a ”preShared Zone” in shared memory until it is finally moved to the ”shared Zone” by tracking its meta data in Zookeeper allowing the transfer of the data’s leadership between nodes.

The Relational-Memory Approach allows native access to a row and column format [37]. Utilizing Field-Programmable Gate Arrays, Programmable Logic In-the-Middle relational operations are implemented in reprogrammable hardware. Queries get then provided with the data by the Relational

Memory Engine in the optimal layout instead of accessing the memory itself.

Finally, partitioning workloads in an intelligent manner without extra statistical data structures is possible, too. As presented by Boissier and Kurzynski [38], physical horizontal data partitioning as well as the adapted aggressive data skipping approach can skip up to 90% of data on OLAP queries.

B. Concurrency

Handling multiple versions of data is a crucial part of all HTAP databases. Current OLTP and OLAP operations require a solution to parallelize data access.

The most common approach is Multi Version Concurrency Control (MVCC). It is utilized in combination with a delta by PostgreSQL [14], SAP HANA [5], F1-Lightning [39], TiDB [18], Poseidon [21] and in new versions of HyPer [40]. To improve scan times in such MVCC systems, a first generic effort is made by vWeaver, implementing a per record frugal skip list to reduce lookups by smartly adding ”shortcut”-pointers between different versions [41].

Hyrise [42] is also using MVCC, but is following a look free commit approach, replacing the delta.

SnappyData and BatchDB [13] also use MVCC oriented approaches. SnappyData [12] relies on GemFire to handle concurrent access and snapshots, while BatchDB [13] uses MVCC on its OLTP replica, while updating the isolated OLAP replica batch-wise.

Although HyPer is now using MVCC with delta, it initially used the fork systemcall to create multiple isolated in-memory snapshots [43]. Utilizing a copy on write approach to reduce memory consumption OLAP queries could be executed on snapshots while the OLTP operations updated the main memory entries.

In addition to the MVCC on its main OLTP replica, SAP HANA [24] further uses ATR with a replication log system to synchronize its multiple server architecture. This synchronizes data with sub-second visibility delay between the replicas.

Instead of using classical MVCC, Polynesia uses a snapshot chain for each column with versions containing whole columns rather than deltas. Following a lazy approach, versions are marked as dirty if changes to the data occur instead of creating a new snapshot. If a dirty column is then accessed by an OLAP query the new snapshot with the latest data is created. These snapshots can further be shared between multiple OLAP queries if required. The row and column store are further synchronized by the update shipping/application unit following two multi component architectures. With this approach simplified transactions are added to queues, intelligently merged to a single update buffer, and then applied to the column store.

Wildfire [25] chooses speed over concurrency as already mentioned. Therefore, a simple last writer wins approach is used by the Wildfire engine.

While most systems nowadays use some implementation of MVCC, research on more efficient snapshotting techniques is still being carried out, e.g., [44]. Inspired by earlier HyPer implementations, another research project on snapshotting, AnKer [45], uses a customized Linux kernel with an updated fork system call. This updated fork, called vm_snapshot, enables high frequency snapshotting. Through vm_snapshot the researchers are able to snapshot only the used columns. This significantly outperforms the default fork used initially by HyPers implementation, providing a possible alternative to MVCC systems.

In addition to PostgreSQL’s locking system, Greenplum further introduces a Global Deadlock Detector (GDD) to resolve deadlocks originating from the distributed server setup [15].

Wait free HTAP (WHTAP) [46] utilizes snapshotting for concurrency as well. In this dual snapshot engine approach data for OLAP and OLTP are stored in different replicas, using a five state process and two deltas. In this process, the deltas form the OLAP and OLTP replicas are switched and the old OLTP delta is merged into the OLAP replica, which takes effect without slowing the analytical queries down.

In VEGITO concurrency can be handled by the epochs utilized to save data [32]. While new updates are always applied to the primary row store of a shard the data is then asynchronously applied to the replica and the analytical column store by a non-volatile write-ahead log batched in epochs. The analytical stores save the latest epoch currently available on all stores and analytical queries are then executed on the epoch, rather than the current epoch used by the primary row store.

AnyDB once again takes a unique approach to concurrency control. By handling the communication between the ACs via event-streams and queues concurrency control is achieved by routing the steams intelligently, merging read requests events on AC nodes with the write requests, only providing the query response after the write event is received [29].

C. Garbage Collection

MVCC implementations require performant garbage collection to prevent large amounts of versions to slow down the transactions on the database. SAP HANA [5] uses timestamps and visibility bits to track versions of their data. Data gets created/edited with a timestamp. When all active transactions can see this version the timestamp is replaced with a bit indicating the visibility. If the row is no longer visible to any snapshot, it can be deleted with the next delta merge.

HyPers garbage collector Steam [40] follows a similar approach. The main difference is that the garbage collector is called with every new transaction instead of being a background task like with SAP HANA. This approach called eager pruning removes all versions not required by any transaction. This happens by checking every time the chain is extended whether all versions included in the version chain are used by a transaction. With eager pruning the version chain can only be as long as the amount of different queries. A similar approach is conducted by Poseidon [21] and Polynesia. In Poseidon on each transaction, data, which is invalid (e.g., due to an aborted transaction) or no longer visible on any version, is pruned from the database. Polynesia deletes all unused snapshots after each

finished analytical query, which are no longer used by any running query.

On VEGITO systems data is saved in epochs rather than with timestamp [32]. As soon as an epoch is outdated (no fresh data is contained in an epoch as all entries are either updated or deleted) the epoch can be removed as a whole from the system.

Due to its heavily distributed architecture with many data sets saved in main memory and SSDs on the different servers, Wildfire follows a different solution and implements a lazy garbage collection approach [25]. When performing lazy garbage collection, data is only deleted if there is no possibility that a query could require it. In Db2 Event Store data is pruned after being moved to the next zone, preventing data duplication [26]. It is also possible to delete the persistent data in the final zone by defining a time to live in the configuration.

D. Query Handling

The ways to access the concurrent data differ significantly from database implementation to implementation.

1) General Query Optimization: Besides the general handling of the queries, Sirin et al. [47] show the importance of isolating the OLTP and OLAP workloads on shared hardware systems in order to achieve optimal performance.

Tested on the Umbra SSD-based HTAP database a worst-case optimal join processing option is introduced providing a general improvement option for HTAP systems [11]. Implementing a hybrid query optimizer a single query plan can be build out of binary and worst-case optimal joins, greatly increasing OLTP performance while having no effect on OLAP workloads.

2) Query Handling in Scale-up Systems: The systems HyPer [48] and Hyrise [16], primarily engineered for scale-up solutions working on one dataset, implemented the query operators as C++ code in their database. The missing variables are inserted via just in time compilation. After the insertion, the code is compiled to LLVM assembler code, allowing fast query execution. As mentioned before, the two databases also have prototype scale-out options, but focus on the scale-up approach.

The LLVM approach is also utilized by Poseidon. The primary commands are already ahead of time (AOT) compiled and the query execution starts instantly. At the same time the query is compiled to LLVM code. If the LLVM code is compiled before the query finishes the query execution is moved from the AOT code to the more optimized LLVM code. The compiled LLVM queries are further saved in the non-volatile memory for faster processing of repetitive queries [21][49].

In Polynesia query operators get arranged in a tree structure [23]. These are then further separated into sub-tasks and if possible executed in parallel, speeding up execution time.

Another approach is to dynamically schedule memory and computing resources actively [50]. Utilizing their algorithm the Resource and Data Exchange engine uses a state based approach to assign the CPU cores to the OLAP or OLTP workload as required, always trying to maximize productivity and the database throughput.

3) Query Handling in Scaled-out Systems: Scale-out systems are separated in two major groups: On the one hand, systems keeping the OLTP workload on one server, scaling only the OLAP workload to other servers, as e.g., SAP HANA [24] or BatchDB [13]. On the other hand, systems distributing OLTP and OLAP workloads over multiple servers, e.g., Wildfire [17][25] and SnappyData [12][28].

BatchDB [13] and HANA [24] both handle their OLTP workload on a single server, scaling-up via NUMA as described earlier. For OLAP workloads they are able to scale out onto multiple servers working on replicas of the main data.

Wildfire [17] and SnappyData [28] contrarily scale out via Apache Spark, allowing OLTP and OLAP transactions to be executed on a cluster of nodes dealing with big data and streaming workloads. Wildfire [17] executes OLTP queries on the fresh data on Wildfire daemons. OLAP workloads can be executed via Spark Executor as requests to the daemons or directly accessing the shared data of the Wildfire database cluster. With this approach, old data can be consumed from the shared file system without slowing down OLTP throughput while the latest data can still be received if required. Db2 Event Store utilizes the Db2 BLU MPP cluster query engine instead of Apache Spark for the sake of better low latency query handling [26].

Greenplum handles all queries in a cluster via its central coordinator [15]. The queries are then processed and handled by workers. Via ”Motion”-nodes data can be transferred between separated machines/segments.

In AnyDB queries can be executed on multiple servers by simply routing the query events to the according ACs [29]. To speed up processing even further data can be ”beamed” to ACs in advance, before the query is available. By determining the AC responsible for the query execution and knowing, which data will be required by the query the data is send to the AC before the query optimization is done, providing a great speed up by parallelizing the two work steps.

4) Query Language: While the databases offer many new functionalities to access and modify data, SQL is still commonly supported. The database systems SAP HANA [8], Wildfire [25], Db2 Event Store [26], Hyrise [16], HyPer [7], SnappyData [12], TiDB [18], AnyDB [29] and AIM [20] all enable basic SQL queries to interact with the database. However, many of them further provide new optimized ways to interact with the data.

Wildfire [25], TiDB [18] and SnappyData [12] provide data access via an extended version of the SparkSQL API. SnappyData also further extends the Spark Streaming API.

SAP HANA [8] provides more specific access through SQL Script and Multidimensional Expressions (MDX). The database is also is natively optimized for the ABAP language and runtime. This allows to bypass the SQL connectivity stack by directly accessing special internal data representations via Fast Data Access (FDA). The Native For All Entries (NFAE)

technique further modifies the ABAP runtime to allow even more performance improvements.

Hyrise [16] provides a command-line interface, which allows SQL queries but also provides additional visualization and management functions. Furthermore, the wire protocol of PostgreSQL allows access through common PostgreSQL drivers and clients.

HyPer [48] uses HyPerScript as its query language. HyPerScript is a SQL-based query language and therefore allows base SQL statements as well. The features consist of passing whole tables as query parameters and providing the possibility to use query results in a later part of the query, removing the need to query the same value multiple times.

E. Indices

To allow efficient data access and querying on multiple servers and/or different versions of data, the right index structure is of special importance in HTAP databases.

Wildfire's multi-version multi-zone index Umzi [51] employs a LSM-like structure with multiple runs. It divides index runs in multiple zones and implements efficient evolve operations to handle zone switches of data. Further Umzi uses a multi-tier storage using SSDs and memory caching with self-updating functionality for fast execution while persisting the indexes on Wildfires shared data. Db2 Event Store's index is based on Umzi [26]. It is only applied to the preShared and shared zone, as the initial zone does not support synchronization.

A general approach to utilizes LSM-Trees for HTAP workloads is further presented by Saxena et al. [52]. Their prototype LASER utilizes a Real-time LSM-Tree allowing to store data in different formats during its lifecycle (similar to Db2 Event Store) providing significant speed ups to traditional methods.

HyPer developed the Adaptive Radix Tree (ART) [48] based on the radix tree. ART uses four different node types that can handle 4, 16, 48 and 256 entries. The maximum height for the tree is k for k-byte trees. To further reduce the tree height and required space, the tree is build lazily, saving single leaf branches higher in the tree. Additionally, path compression is used to remove common paths and to insert them as a prefix of the inner node thereby removing cache inefficient one-way node chains.

SAP HANA [5] and Hyrise [16] both use B-Trees. Hyrise further supports the ART index from HyPer [48] and a group-key-index, implemented by the Hyrise project.

TiDB uses its own implementation, the DeltaTree [18]. Its creators further built a B+-Tree on top of the delta tree, to speed up updates on key ranges and look ups on single key values, as well as merging of deltas.

A B+-Tree is further used as the index of Poseidon [21]. The leafs of the B+-Tree are saved in non-volatile memory, providing persistency and recovery options, while the inner structure of the tree is saved in memory. With this approach only one non main memory access has to be conducted, while still providing enough persistency for efficient recovery.

Once again utilizing its epoch system VEGITO [32] can use a buffered tree based index. As changes to the OLAP index tree only must be applied after an epoch finishes all changes during an epoch are applied in parallel to buffers and the three weights are updated. In the update step at the end of an epoch the tree first gets optimized (also a split is possible here) and the the buffered changes are applied. This approach evades slowdowns by removing the need for locking with buffers.

BatchDB utilizes a simplified version of the look-free Bw-Tree [13]. The version relies on atomic multi-word compare-and-swap updates.

In 2019, a predictive indexing approach [53] was introduced to cope with the dynamic demands of a HTAP database. Predictive indexing increases the throughput by up to 5%. In this approach, a machine learning system calculates the optimal index structure for the data according to the workload. A similar approach can be seen in the Multi-armed bandit solution from 2021 [54]. With this approach even greater improvements up to 59% speedup are possible.

The Multi-Version Partitioned B-Tree (MV-MBT) [55] is another recent research in the indexing sector for HTAP databases from 2019. This extension of partitioned B-Tree creates a version aware index, able to maintain multiple partitions within a single tree structure, sorted in alphanumeric order.

Likewise proposed in 2019, the Parallel Binary Tree (P-Tree) [56] is an extension of a balanced binary tree relying on copy-on write mechanisms to create tree copies on updates. With this approach, the indices become the version history without requiring other data structures.

F. Big Data on HTAP Databases

Wildfire/Db2 Event Store was created with big data IoT workloads as its primary use case [25][26]. Through the distributed design of its big data platform, Wildfire is able to concurrently handle high-volumes of transactions as well as execute analytics on latest data. At the same time, the system is able to scale onto many machines because of its close integration of Apache Spark. The usage of an open data format further enables compatibility with the big data ecosystem. Nowadays, the commercial version IBM Db2 Event Store is capable of handling more than 250 billion events per day [57][26]. SnappyData [28] is an analogically capable big data platform with an architecture similar to Wildfire.

The SAP HANA database can be used as part of the SAP HANA data platform to handle big data workloads [58]. Using a combination of different SAP products, namely SAP Synbase ESP and SAP Synbase IQ, as well as smart data access frameworks as Hadoop, Teradata or Apache Spark, the SAP HANA data platform is a fully functional big data system with SAP HANA in its core.

HyPerInsight [59] provides big data capabilities in the area of data exploration on the HyPer database. The goal is to minimize the required user expertise with the dataset while simultaneously supporting the user with the formulation of queries. The support for lambda functions in SQL queries

allows user defined code to be executed within the queries. In combination with the HTAP HyPer system as the database, data mining on real-time data is possible.

G. Recovery, Error Handling and Logging

As many HTAP databases rely on volatile main memory as primary storage and the other systems utilize distributed data sets, recovery in case of failure is of special importance. Data loss has to be prevented and downtime must be minimized.

SAP HANA instances log data persistently on the local drive for recovery on failure or restart purposes [8]. The logging approach is inspired by SAP MaxDB.

As already explained, HANA works with ATR in its distributed architecture [24]. Following the store-and-forward approach, the data is replicated to multiple servers. An algorithm then compares the record version IDs of the incoming data and stored data, requesting the resend of lost log entries if deviations occur.

Recovery for the latest version of Hyrise is still work in progress [16], but recovery for older versions of Hyrise was explained [42]. The database dumps the main partition of the table as a binary dump on the disk and records the delta to a log via group commits to hide the latency. At checkpoints, the delta partitions are also saved as a binary dump on the drive. If recovery is required, the main dump and delta dump from a checkpoint are restored and an eventually existing delta log is replayed on the table, restoring the old state.

BatchDB logs successful transactions on its OLTP replica in batches via command logging on durable storage [13]. In case of a failure, the database can recover from these logs. The OLAP replica itself has no durable logging and has to recover from the main OLTP replica on failure.

SnappyData [12] uses Apache Sparks logging and recovery mechanisms, logging transformations used to build Sparks Resilient Distributed Datasets (RDDs). Saving RDDs to storage is also possible. In SnappyData the combination with GemFire however allows Spark to save the RDDs in GemFires storage instead of the persistent storage of the server. Small recoveries can be handled directly by GemFires eager replication, leaving batched and streaming recovery to Spark, in combination with the GemFire storage. Further, a peer-to-peer (p2p) approach is used in SnappyData clusters. Any in-memory data can be synchronously replicated from the cluster. Additional to the replication via the p2p approach, data is always replicated to at least one other node in the cluster.

VEGITO [32] offers a quite refined recovery system, handling four failure cases. If the primary row store fails the leader role is transferred to its backup row store and a new backup row store is created. In case of a backup store failure it is simply recreated from the primary row store. If the OLAP column store fails it can be recreated from the row stores. Finally, if both row stores fail it is possible to recreate the primary row store on the same machine as the column store, afterwards a backup row store is initialized.

TiDB Raft groups recover in case of a leader failure by electing a new leader from the groups followers. OLTP workloads are then simply redirected to the new leader and continue there [18].

AnyDB once again introduces a new approach. As all communication inside AnyDB is handled by event streams these can simply be rerouted to a functional AC in case of a failure of an AC/server [29].

IBM's Db2 Event Store provides a catalog, which contains meta data required for initial cache population [26]. In order to prevent possible data loss the catalog data is saved in a shared persistent storage and provided to the system via a logical node. If the catalog node fails, the data can be reloaded from this storage by another server of the cluster, which then resumes the work of the catalog node. All data is further saved on fast local storage of one node, as well as to the local file system of at least two other nodes, allowing for easy recovery.

H. Benchmarking

The combination of OLTP and OLAP workloads on one database also created the need for new benchmarks covering this sector. In 2011, CH-benCHmark [60] was introduced. The CH-benCHmark is based on the TPC-C and TPC-H benchmarks. It executes a transactional and analytical workload in parallel on a shared set of tables on the same database. The benchmark can also be used for single workload databases.

In 2017, HTAPBench [61] was published. This benchmark is able to compare OLTP, OLAP and hybrid workloads on the database. Its main difference to CH-benCHmark lies in its Client Balancer, controlling the coexisting OLAP and OLTP workloads.

Hyrise [16] implements a special benchmark runner to easily execute benchmarks.

Another benchmark specially designed for document oriented NoSQL platforms is introduced by Tian et al. [62].

I. Stream Processing

Streaming as a special case of OLTP is an emerging use case for HTAP database systems. In 2016 scientists from ETH Zürich in cooperation with Huawei presented AIM [20], which is a high performance event-processing and real time analytics HTAP database. The three-tiered multi node system processes events at one tier, stores the data at a central tier and finally analyses the processed data in real time on the third tier. AIM however, is optimized for a special streaming use case from the telecommunications industry.

In early 2019, the research team around HyPer compared modified versions of HyPer with AIM and Apache Flink [63] in order to determine the current state of streaming capabilities of main memory database systems (MMDB). While MMDBs are still inferior to dedicated streaming frameworks like Flink, the HyPer team was confident, that HTAP databases could catch up with some adjustments, even implementing some of those on HyPer. The main areas requiring improvement are network optimization, parallel transaction processing, skew handling and a strong distributed architecture.

SnappyData aims to solve OLTP, OLAP and streaming all in one product [28] with their tight integration of Apache

Spark and GemFire. In an evaluation, SnappyData was able to outperform both Spark on TPC-H queries as well as MemSQL on all kinds of throughput. The focus with SnappyDatas stream processing lies on complex analytical queries on streams, which are not possible with default stream processor solutions [12].

SAPs approach for big data, SAP Big Data [58], supports streaming as well. Since it uses other SAP products to achieve it and is not a part of the base SAP HANA database infrastructure, but rather is built on top of it, it is not further discussed in this paper.

J. HTAP as a Service

In the last year HTAP became more and more popular with cloud solutions. Following this trend some of the global players enabled "HTAP-as-a-service" (HaaS) for their existing could solutions. Instead of providing new databases they introduce HTAP-like functionality with tightly coupled OLTP and OLAP multi database setups. Still they provide some interesting new research and provide a valid solution for production systems.

1) F1-Lightning: F1-Lightning consists of distributed multi actor system with data aggregation components, in memory and on disk data storage, as well as a metadata database, containing all required information to function. Instead of being a fully functional HTAP database itself, it can be selected as an addition to the Google Cloud Platform databases F1-DB and Spanner. Either for some tables or whole databases the analytical queries are moved to the F1-Lightning cluster, while the transactional queries stay on the OLTP databases. With this approach the currently used database does not have to be exchanged and F1-Lightning can be added and removed on the fly. While currently only F1-DB and Spanner are supported as the base OLTP database the architecture is highly adjustable and further databases could be supported in the future. For the databases currently being extended F1-Query is utilized as a query language.

Using a MVCC approach with snapshot isolation the queries are conducted either in the OLAP improved Lightning tables or on the OLAP F1-DB/Spanner tables, if the data was not yet copied. Using a time-to-live (TTL) based garbage collection approach and compaction techniques older data is deleted and small deltas are combined to larger, more memory efficient blocks. However, the used garbage collection approach also deletes still valid data, if it was not changed in a certain time span.

Due to information stored in the metadata database and the modular architecture, partitions of data can be split and merged on the fly as required, without disturbing the analytical queries. If certain deltas getting to large for in-memory usage they are moved and transformed from the row based in memory store to on-disk storage in a read optimized column structure.

2) Azure Synapse Link: Azure Synapse Link is currently only available as a service for Azure Cosmos DB [64]. Unlike F1-Lightning it can only be added to the whole database. Moreover, there is a high delay between the latest OLTP data and the data used for OLAP queries of 2-5 minutes. In its core functionality Azure Synapse Link synchronizes the data from the row based Azure Cosmos DB to a column based analytical database and provides a bridge to Azure Synapse Analytics. Like F1-Lightning a TTL-based garbage collection is utilized.

K. Future

HTAP databases are a new sector, which has evolved over the past 10 years. On an annual basis, companies and researchers contribute new ideas to lift their database above the competition. While we stated in our last work, that there are currently three trends it seems like a fourth has emerged.

1. CPU GPU collaboration: With new hardware supporting heterogeneous parallelism, Heterogeneous HTAP (HHTAP/H2TAP) becomes a possibility. In this approach, CPUs and GPGPUs can access shared memory and divide the workload between both. Complex OLAP queries are solved on the GPGPUs, leaving the OLTP workload for the CPUs. The Caldera [65] prototype proved the feasibility for HHTAP. Early 2020, the data store GridTables [66] was published and affirmed the concept again. However, in their summary, the authors of GridTables pointed out that there are still many research issues left to be solved. Further, early in 2020, a paper was published about GPU accelerated data management [67] explaining how to fully exploit hardware isolation between CPUs and GPUs and presenting a SemiLazy access method to reduce the required data transfer. EEVEE (inspired by and named after the highly adaptable Pokémon Evoli/Eevee) provides an interesting approach for scheduling such HHTAP workloads [68]. While the work was never formally published it can provide some valuable insights for upcoming developments in this area.

2. Streaming workloads: as described in detail in the previous section, stream processing is a possible use case for HTAP databases. Because of the optimization for high OLTP throughput and the ability to analyse these data streams in the same system, HTAP databases are an emerging alternative to current stream processing solutions. While still inferior to dedicated stream processors, the research on such solutions saw an increase in interest over the last years, e.g., by the HyPer team [63] and dedicated streaming HTAP databases like SnappyData [28] and AIM [20].

3. Optimization: while the bigger part of the last decade was spent on researching for new systems [7][8][42], the last quarter focused on their optimization. Few new database systems were proposed and research started optimizing existing systems even further [40][45][56]. Also research focusing on the general improvement of HTAP systems becomes more common, e.g., [47][41][52].

4. Over the last year the interest in NVMe storage has greatly increased. Different research is investigating how to efficiently utilize this kind of non volatile memory [21][35] while production systems already started utilizing it [26]. While using NVMe instead of ordinary SSD may not seem that interesting at first this approach offers many new opportunities (mainly connected to way better speed than ordinary SSDs

while much cheaper than main memory) and challenges arise (e.g., problems with upgrading traditional disk-based systems to NVMe).

Further, solutions utilizing machine learning in combination with HTAP are slowly emerging. These allow databases to adapt on their own according to current workload and requirements. However, there is still not enough research to speak of an own trend and it can rather be viewed as another kind of optimization research. An example for such research is the presented predictive indexing [53] or the Multi-armed bandit approach [54].

Another new development is the integration of HTAP capabilities into polystore databases. While polystore and HTAP databases evolved mostly disjunct over the last years first approaches to combine both paradigms were conducted with Polypheny-DB [69] and [70].

Not being a trend in the HTAP database development, because simply not being HTAP databases, more and more solutions emerge providing HTAP capabilities to pre-existing database systems, efficiently bridging the gap between OLTP and OLAP systems instead of creating new combined database systems. Examples can be found by the HTAP-as-a-Service solutions [39][64], as well as systems like IBM Db2 Analytics Accelerator [71].

L. Open Source and Free Versions

Some of the database systems summarized in this paper provide open source and/or free solutions.

SnappyData [76] is available with a getting started guide covering the basic usage. The source code can be found on GitLab. The project is licensed with the Apache License, Version 2.0. A comment in the GitHub Readme however declares the project as legacy. An official statement to the state of SnappyData could not be found.

MemSQL [77] (now SingleStore), is available, well documented and can be used for smaller projects up to 4 nodes for free. Many extensions are available at the official GitHub account.

Hyrise [75] is available under the MIT license. However, as it is a research database, breaking changes may occur more frequently.

A initial version of VEGITO is available under the Apache License Version 2.0 on GitHub [78]. With the same license a production ready version of TiDB is also available on GitHub [72]. It is provided with a complete documentation, however, 32 GB+ RAM are advised for a minimal production setup and enabling the HTAP capabilities requires further setup and understanding of some of the underlying software solutions.

Poseidon is available via GitHub under the GPL 3.0 license [79]. However, while the publications to this database are from 2021 the last commit is from February 2020. The git repository therefore seems outdated.

While PostgreSQL itself is available open source and can be modified to handle HTAP workloads [80] there are some other open source database systems building on top of PostgreSQL. These are most of the time more refined and easier to use with HTAP workloads. Greenplum (Apache 2.0 License) [73] requires linux servers with 16 GB RAM and all utilized servers of a Greenplum cluster require the same hard- and software configuration.

The free MemSQL version and a basic SnappyData setup can already be used on low end systems, naming 8GB main memory as their minimal requirement to operate efficiently.

To try a HTAP database without a setup process, HyPer and Umbra can be used. Both research version are provided via a simple web tool for exploration and testing [81] [82]. This version, however, is running on a low end system and cannot be used in production.

V. Production Ready HTAP Databases

While the previous part of this work is solely based on the systematic literature review this section also highlights databases, which were not found during the review process, if they are a valid production ready solution. The focus lays on technologies already mentioned in the literature review part of this paper, which are production ready, as well as the big players. Besides HTAP databases we further include HaaS solutions, enabling HTAP on existing OLTP cloud systems. Table I provides a compact overview (only links to free versions are provided).

VI. Conclusion

In this paper, we have shown that HTAP databases are nowadays serious alternatives to traditional database solutions. The existence of a market for commercial products like SAP HANA, IBM Db2 Event Store and Tableau further reinforces our findings. Moreover, we have highlighted differences of existing approaches regarding key properties like the fundamental architecture, concurrency, or big data capabilities. Furthermore, we have highlighted a set of currently available production ready HTAP implementations ranging from open source systems to commercial products. Thus, this study can aid both researchers and practitioners in the process of selecting a matching HTAP solution. Finally, by providing a comprehensive overview of current approaches, this study helps to identify trends and point out directions for future research. As there is a great amount of active research in this area, this article builds upon our previous literature study on this topic and enhances it with new alternatives as well as production ready approaches. The following paragraphs provide a brief summary of our findings.

Open source and free HTAP products place HTAP databases on the same level as traditional database systems, allowing the integration in other products and exploring this new technology without financial risks.

The combination of OLTP and OLAP queries on one database efficiently reduces the total cost of ownership and allows a narrower tech stack for companies. The possibility to analyse data in real time further validates HTAP databases as a productive solution with a great added value compared to conventional databases.

TABLE I
PRODUCTIVE HTAP DATABASE SYSTEMS

Database	Availability	Pricing	Core Facts
F1-Lightning	- Google Cloud Platform	- Spanner subscription + - F1-DB subscription +	highly scalable; can be added and removed on the fly; extends existing OLTP databases; (if already using F1-DB or Spanner) no initial database migration required; only available for Spanner and F1-DB; no "real" HTAP
Azure Synapse Link	- Azure	- requires multiple Azure subscriptions	can be added on the fly; extends existing OLTP databases; no initial database migration required; only available for Azure Cosmos DB; no "real" HTAP; high delay between OLTP and OLAP store (2-5 minutes)
IBM Db2 Event Store	- local - own Linux cluster - IBM Cloud Pak	- free developer version - free test, not specified - not specified	enormous scalability; IoT optimized; free developer version; can be self hosted (in contrast to Azure/Google Cloud); deeply integrated with other IBM solutions; very active ongoing research
SAP HANA	- self hosted - cloud	- free test, not specified	highly integrated with other SAP products; very active ongoing research; distributed or single server scaling
IBM Db2 Analytics Accelerator	- on IBM z/OS systems	- not specified	HTAP database extension for Db2 for z/OS systems; tightly coupled with other IBM products; requires a Db2 for z/OS database on a mainframe; very active ongoing research; one of the most researched systems
Amazon Aurora	- AWS	- multiple AWS subscriptions	cloud only; low visibility in HTAP research; easy scalability
Tableau	- hosted - self hosted	- stating at 12$/month	based on HyPer; one of the most researched systems; part of tableau technology stack; not available as simple database
TiDB	- GitHub [72]	- open source - enterprise available	highly scalable; moderate minimal hardware requirements; high fault resistance; setup requires technical knowledge
Greenplum	- GitHub [73]	- open source	PostgreSQL based; highly scalable; moderate minimal hardware requirements; integrated machine learning and analytical capabilities
Swarm64	- self hosted - cloud	- free test version - starting 528$/month	PostgreSQL based; can be used on common clouds; not present in research
Citus	- GitHub [74]	- open source	PostgreSQL based; great scalability; available on Azure; some research papers; multi-node PostgreSQL cluster
Hyrise	- GitHub [75]	- open source	very active research; active development; research system not suitable for production

Many different implementations, providing different advantages, are available and can be used as required by the costumer. Solutions using main memory as a primary/sole storage as well as solutions relying on shared data storages exist and are both valid options. Powerful single server database systems allow a slim tech stack while still being faster than most traditional OLTP and OLAP optimized databases. Distributed multi server clusters allow more fail-safe and easier to scale solutions, while at the same time requiring less performant machines. SnappyData and MemSQL, for example, can already be executed on machines with 8GB of memory, scaling up from there.

Over the last years, new indices, filters, data structures and replication techniques were developed, optimizing performant HTAP systems even further. The future seems to be heading in three main directions: HHTAP - utilizing new heterogeneous hardware to include the GPUs in HTAP databases and allow even more efficient architectures.

Streaming - HTAP databases optimized for streaming are making a combination with external stream processors unnecessary, further reducing the total cost of ownership and reducing the size of the required tech stack.

Optimization - while the bigger part of the 2010's was spent on developing the base technologies and databases themselves, the last quarter was primarily spent on optimization, still leaving much room for improvement.

Machine learning for self adapting databases also could be an emerging sector in the future, but currently there is not enough research in this direction to call it a trend.

There seems to be a growing interest in HTAP solutions yielding high numbers of novel approaches and providing innovative solutions regrading many technical aspects of databases. Therefore, our future work will focus on further exploring the state of research. Due to the amount of active research it will be difficult to capture the whole area in a comprehensive literature review. Thus, we aim at providing in-depth overviews of different aspects of this database area.

REFERENCES

[1] D. Hieber and G. Grambow, "Hybrid transactional and analytical processing databases: A systematic literature review," *DATA ANALYTICS 2020, The Ninth International Conference on Data Analytics*, pp. 90–98, 2020.

[2] R. Nigel, F. Donald, P. Massimo, and E. Roxane, "Hybrid transaction/analytical processing will foster opportunities for dramatic business innovation," 2014, last visited: 12.06.2022. [Online]. Available: https://www.gartner.com/en/documents/2657815

[3] U. Joseph *et al.*, "Predicts 2016: In-memory computing-enabled hybrid transaction/analytical processing supports dramatic digital business innovation," 2015, last visited: 12.06.2022. [Online]. Available: https://www.gartner.com/en/documents/3179439

[4] Y. Noel and G. Mike, "Emerging technology: Translytical databases deliver analytics at the speed of transactions," 2015, last visited: 12.06.2022. [Online]. Available: https://www.forrester.com/report/Emerging-Technology-Translytical-Databases-Deliver-Analytics-At-The-Speed-Of-Transactions/RES116487

[5] N. May, A. Böhm, and W. Lehner, "Sap hana – the evolution of an in-memory dbms from pure olap processing towards mixed workloads," in *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, T. Härder, O. Kopp, and M. Wieland, Eds. Gesellschaft für Informatik, Bonn, 2017, pp. 545–546.

[6] A. Kemper, V. Leis, and T. Neumann, *Die Evolution des Hauptspeicher-Datenbanksystems HyPer: Von Transaktionen und Analytik zu Big Data sowie von der Forschung zum Technologietransfer*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 149–154. ISBN 978-3-662-54712-0

[7] A. Kemper and T. Neumann, "Hyper: A hybrid oltp olap main memory database system based on virtual memory snapshots," in *2011 IEEE 27th International Conference on Data Engineering*, 2011, pp. 195–206.

[8] F. Färber *et al.*, "The sap hana database - an architecture overview," *Bulletin of the Technical Committee on Data Engineering / IEEE Computer Society*, vol. 35, no. 1, pp. 28–33, 2012.

[9] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, 08 2004.

[10] T. Neumann and M. J. Freitag, "Umbra: A disk-based system with in-memory performance," in *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org, 2020. [Online]. Available: http://cidrdb.org/cidr2020/papers/p29-neumann-cidr20.pdf

[11] M. J. Freitag, M. Bandle, T. Schmidt, A. Kemper, and T. Neumann, "Adopting worst-case optimal joins in relational database systems," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1891–1904, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1891-freitag.pdf

[12] B. Mozafari, "Snappydata," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019.

[13] D. Makreshanski, J. Giceva, C. Barthels, and G. Alonso, "Batchdb: Efficient isolated execution of hybrid oltp+olap workloads for interactive applications," *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.

[14] M. Nakamura *et al.*, "Extending postgresql to handle olxp workloads," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 40–44.

[15] Z. Lyu, H. H. Zhang, G. Xiong, G. Guo, H. Wang, J. Chen, A. Praveen, Y. Yang, X. Gao, A. Wang, W. Lin, A. Agrawal, J. Yang, H. Wu, X. Li, F. Guo, J. Wu, J. Zhang, and V. Raghavan, *Greenplum: A Hybrid Database for Transactional and Analytical Workloads*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 2530–2542. ISBN 9781450383431

[16] M. Dreseler *et al.*, "Hyrise re-engineered: An extensible database system for research in relational in-memory data management," in *EDBT*, 2019.

[17] R. Barber *et al.*, "Evolving databases for new-gen big data applications," in *CIDR*, 2017.

[18] D. Huang, Q. Liu, Q. Cui, Z. Fang, X. Ma, F. Xu, L. Shen, L. Tang, Y. Zhou, M. Huang, W. Wei, C. Liu, J. Zhang, J. Li, X. Wu, L. Song, R. Sun, S. Yu, L. Zhao, N. Cameron, L. Pei, and X. Tang, "Tidb: A raft-based htap database," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3072–3084, Aug. 2020.

[19] V. Arora, F. Nawab, D. Agrawal, and A. E. Abbadi, "Janus: A hybrid scalable multi-representation cloud datastore," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 689–702, 2018.

[20] L. Braun *et al.*, "Analytics in motion: High performance event-processing and real-time analytics in the same database," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2723372.2742783. ISBN 9781450327589 p. 251–264. [Online]. Available: https://doi.org/10.1145/2723372.2742783

[21] M. Jibril, A. Baumstark, P. Götze, and K.-U. Sattler, "Jit happens: Transactional graph processing in persistent memory meets just-in-time compilation," in *24th International Conference on Extending Database Technology (EDBT)*, 03 2021. doi: 10.5441/002/edbt.2021.05 pp. 37–48.

[22] T. Mühlbauer, W. Rödiger, A. Reiser, A. Kemper, and T. Neumann, "Scyper: elastic olap throughput on transactional data," in *DanaC '13*, 2013.

[23] A. Boroumand, S. Ghose, G. F. Oliveira, and O. Mutlu, "Polynesia: Enabling effective hybrid transactional/analytical databases with specialized hardware/software co-design," *CoRR*, vol. abs/2103.00798, 2021. [Online]. Available: https://arxiv.org/abs/2103.00798

[24] J. Lee *et al.*, "Parallel replication across formats in sap hana for scaling out mixed oltp/olap workloads," *Proc. VLDB Endow.*, vol. 10, no. 12, p. 1598–1609, Aug. 2017.

[25] R. Barber, V. Raman, R. Sidle, Y. Tian, and P. Tözün, *Wildfire: HTAP for Big Data*. Germany: Springer, 2019.

[26] C. Garcia-Arellano, H. Roumani, R. Sidle, J. Tiefenbach, K. Rakopoulos, I. Sayyid, A. Storm, R. Barber, F. Ozcan, D. Zilio, A. Cheung, G. Gershinsky, H. Pirahesh, D. Kalmuk, Y. Tian, M. Spilchen, L. Pham, D. Pepper, and G. Lushi, "Db2 event store: A purpose-built iot database engine," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3299–3312, aug 2020.

[27] R. Barber *et al.*, "Wiser: A highly available HTAP DBMS for iot applications," in *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019*. IEEE, 2019. doi: 10.1109/BigData47090.2019.9006519 pp. 268–277.

[28] R. Jags *et al.*, "Snappydata: Streaming, transactions, and interactive analytics in a unified engine," ser. SIGMOD '16, 2016. ISBN 0-89791-88-6/97/05

[29] T. Bang, N. May, I. Petrov, and C. Binnig, "Anydb: An architecture-less dbms for any workload," in *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org, 2021. [Online]. Available: http://cidrdb.org/cidr2021/papers/cidr2021_paper10.pdf

[30] H. Lang *et al.*, "Data blocks: Hybrid oltp and olap on compressed storage using both vectorization and compilation," in *SIGMOD '16*, 2016.

[31] M. Boissier, R. Schlosser, and M. Uflacker, "Hybrid data layouts for tiered htap databases with pareto-optimal data placements," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 209–220.

[32] S. Shen, R. Chen, H. Chen, and B. Zang, "Retrofitting high availability mechanism to tame hybrid transaction/analytical processing," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, Jul. 2021. ISBN 978-1-939133-22-9 pp. 219–238.

[33] M. Athanassoulis, K. S. Bøgh, and S. Idreos, "Optimal column layout for hybrid workloads," *Proc. VLDB Endow.*, vol. 12, no. 13, pp. 2393–2407, Sep. 2019.

[34] J. Arulraj, A. Pavlo, and P. Menon, "Bridging the archipelago between row-stores and column-stores for hybrid workloads," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2882903.2915231. ISBN 9781450335317 p. 583–598.

[35] G. Haas, M. Haubenschild, and V. Leis, "Exploiting directly-attached nvme arrays in dbms," in *CIDR*, 2020.

[36] P. R. P. Amora, E. M. Teixeira, F. D. B. S. Praciano, and J. C. Machado, "Smartltm: Smart larger-than-memory storage for hybrid database systems," in *SBBD*, 2018.

[37] S. Roozkhosh, D. Hoornaert, J. H. Mun, T. I. Papon, U. Drepper, R. Mancuso, and M. Athanassoulis, "Relational memory: Native in-memory accesses on rows and columns," *CoRR*, vol. abs/2109.14349, 2021. [Online]. Available: https://arxiv.org/abs/2109.14349

[38] M. Boissier and D. Kurzynski, "Workload-driven horizontal partitioning and pruning for large htap systems," in *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*, 2018, pp. 116–121.

[39] J. Yang, I. Rae, J. Xu, J. Shute, Z. Yuan, K. Lau, Q. Zeng, X. Zhao, J. Ma, Z. Chen, Y. Gao, Q. Dong, J. Zhou, J. Wood, G. Graefe, J. Naughton, and J. Cieslewicz, "F1 lightning: Htap as a service," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3313–3325, aug 2020.

[40] J. Böttcher, V. Leis, T. Neumann, and A. Kemper, "Scalable garbage collection for in-memory mvcc systems," *Proc. VLDB Endow.*, vol. 13, no. 2, p. 128–141, Oct. 2019.

[41] J. Kim, K. Kim, H. Cho, J. Yu, S. Kang, and H. Jung, *Rethink the Scan in MVCC Databases*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 938–950. ISBN 9781450383431

[42] D. Schwalb, M. Faust, J. Wust, M. Grund, and H. Plattner, "Efficient transaction processing for hyrise in mixed workload environments," in *IMDM@VLDB*, 2014.

[43] F. Funke, A. Kemper, T. Mühlbauer, T. Neumann, and V. Leis, "Hyper beyond software: Exploiting modern hardware for main-memory database systems," *Datenbank-Spektrum*, vol. 14, no. 3, pp. 173–181, 2014.

[44] L. Li *et al.*, "A comparative study of consistent snapshot algorithms for main-memory database systems," *ArXiv*, vol. abs/1810.04915, 2018.

[45] A. Sharma, F. M. Schuhknecht, and J. Dittrich, "Accelerating analytical processing in mvcc using fine-granular high-frequency virtual snapshotting," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3183713.3196904. ISBN 9781450347037 p. 245–258.

[46] L. Li, G. Wu, G. Wang, and Y. Yuan, "Accelerating hybrid transactional/analytical processing using consistent dual-snapshot," in *Database Systems for Advanced Applications*. Cham: Springer International Publishing, 2019. ISBN 978-3-030-18576-3 pp. 52–69.

[47] U. Sirin, S. Dwarkadas, and A. Ailamaki, "Performance characterization of HTAP workloads," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, Apr. 2021. doi: 10.1109/icde51399.2021.00162

[48] K. Alfons *et al.*, "Transaction processing in the hybrid oltp&olap main-memory database system hyper," *IEEE Computer Society Data Engineering Bulletin*, vol. Special Issue on "Main Memory Databases", 2013.

[49] A. Baumstark, M. A. Jibril, and K.-U. Sattler, "Adaptive query compilation in graph databases," in *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, apr 2021. doi: 10.1109/icdew53142.2021.00027

[50] A. Raza, P. Chrysogelos, A. G. Anadiotis, and A. Ailamaki, "Adaptive HTAP through elastic resource scheduling," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020. doi: 10.1145/3318464.3389783 pp. 2043–2054.

[51] C. Luo *et al.*, "Umzi: Unified multi-zone indexing for large-scale htap," in *EDBT*, 2019.

[52] H. Saxena, L. Golab, S. Idreos, and I. F. Ilyas, "Real-time lsm-trees for HTAP workloads," *CoRR*, vol. abs/2101.06801, 2021. [Online]. Available: https://arxiv.org/abs/2101.06801

[53] J. Arulraj, R. Xian, L. Ma, and A. Pavlo, "Predictive indexing." arXiv, 2019. doi: 10.48550/ARXIV.1901.07064

[54] R. M. Perera, B. Oetomo, B. I. P. Rubinstein, and R. Borovica-Gajic, "No dba? no regret! multi-armed bandits for index tuning of analytical and HTAP workloads with provable guarantees," *CoRR*, vol. abs/2108.10130, 2021. [Online]. Available: https://arxiv.org/abs/2108.10130

[55] C. Riegger, T. Vinçon, R. Gottstein, and I. Petrov, "Mv-pbt: Multi-version index for large datasets and htap workloads," *ArXiv*, vol. abs/1910.08023, 2020.

[56] Y. Sun, G. Blelloch, W. S. Lim, and A. Pavlo, "On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes," *Proc. VLDB Endow.*, vol. 13, pp. 211–225, 2019.

[57] "Ibm db2 event store," last visited: 12.10.2020. [Online]. Available: https://www.ibm.com/de-de/products/db2-event-store

[58] N. May *et al.*, "Sap hana - from relational olap database to big data infrastructure," in *EDBT*, 2015.

[59] N. Hubig *et al.*, "Hyperinsight: Data exploration deep inside hyper," *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

[60] R. L. Cole *et al.*, "The mixed workload ch-benchmark," in *DBTest '11*, 2011.

[61] F. Coelho, J. a. Paulo, R. Vilaça, J. Pereira, and R. Oliveira, "Htapbench: Hybrid transactional and analytical processing benchmark," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3030207.3030228. ISBN 9781450344043 p. 293–304.

[62] Y. Tian, M. Carey, and I. Maxon, "Benchmarking hoap for scalable document data management: A first step," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020. doi: 10.1109/BigData50022.2020.9377937 pp. 2833–2842.

[63] A. Kipf *et al.*, "Scalable analytics on fast data," *ACM Trans. Database Syst.*, vol. 44, no. 1, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3283811

[64] B. Shiyal, "Chapter 9 - synapse link," in *Beginning Azure synapse analytics transition from data warehouse to data lakehouse*. S.l: Apress, 2021. ISBN 978-1-4842-7060-8

[65] A. Raja, K. Manos, P. Danica, and A. Anastasia, "The case for heterogeneous htap," *8th Binnial conference on Innovative Data Systems Reseach (CIDR '17)*, 2017.

[66] M. Pinnecke, G. Campero Durand, D. Broneske, R. Zoun, and G. Saake, "Gridtables: A one-size-fits-most h2tap data store: Vision and concept," *Datenbank-Spektrum*, 01 2020.

[67] A. Raza, P. Chrysogelos, P. Sioulas, V. Indjic, A. C. Anadiotis, and A. Ailamaki, "Gpu-accelerated data management under the test of time," in *CIDR*. Zenodo, Jan. 2020. doi: 10.5281/zenodo.3827490

[68] K. Agrawal, A. Balasubramanian, S. Kamat, and G. P. M. Krishnan, "Scheduling for htap systems on cpu-gpu clusters," 2020. [Online]. Available: https://arjunbala.github.io/wisc-cs839-ngdb20-paper177.pdf

[69] M. Vogt, N. Hansen, J. Schönholz, D. Lengweiler, I. Geissmann, S. Philipp, A. Stiemer, and H. Schuldt, "Polypheny-db: Towards bridging the gap between polystores and htap systems," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer International Publishing, 2021. ISBN 978-3-030-71055-2 pp. 25–36.

[70] P. Kranas, B. Kolev, O. Levchenko, E. Pacitti, P. Valduriez, R. Jiménez-Peris, and M. Patiño-Martinez, "Parallel query processing in a polystore," *Distributed and Parallel Databases*, vol. 39, no. 4, pp. 939–977, Feb. 2021.

[71] D. Butterstein, D. Martin, K. Stolze, F. Beier, J. Zhong, and L. Wang, "Replication at the speed of change: A fast, scalable replication solution for near real-time htap processing," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3245–3257, aug 2020.

[72] "Tidb - github repository," last visited: 30.12.2021. [Online]. Available: https://github.com/pingcap/tidb

[73] "Greenplum - github repository," last visited: 31.12.2021. [Online]. Available: https://github.com/greenplum-db/gpdb

[74] "Citus - github repository," last visited: 02.01.2022. [Online]. Available: https://github.com/citusdata/citus

[75] "Hyrise github," last visited: 12.10.2020. [Online]. Available: https://github.com/hyrise/hyrise

[76] SnappyData. Snappydata 1.2.0 - getting started in 5 minutes or less. Last visited: 12.10.2020. [Online]. Available: https://snappydatainc.github.io/snappydata/quickstart/

[77] MemSQL. Memsql documentation. Last visited: 12.10.2020. [Online]. Available: https://docs.memsql.com/v7.1/introduction/documentation-overview/

[78] "Vegito - github repository," last visited: 30.12.2021. [Online]. Available: https://github.com/SJTU-IPADS/vegito

[79] "Poseidon - github repository," last visited: 31.12.2021. [Online]. Available: https://github.com/dbis-ilm/poseidon_core

[80] "Postgresql - github repository," last visited: 31.12.2021. [Online]. Available: https://github.com/postgres/postgres

[81] "Hyper online interface," last visited: 12.10.2020. [Online]. Available: http://hyper-db.de/interface.html

[82] "Umbra online interface," last visited: 01.01.2022. [Online]. Available: https://umbra-db.com/interface/

A Hybrid Graph Analysis and Machine Learning Approach Towards Automatic Software Design Pattern Recognition Across Multiple Programming Languages

Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
e-mail: roy.oberhauser@hs-aalen.de

***Abstract*—The volume of program source code created, reused, and maintained worldwide is rapidly increasing, yet code comprehension remains a limiting productivity factor. For developers and maintainers, well known common software design patterns and the abstractions they offer can help support program comprehension. However, manual pattern documentation techniques in code and code-related assets such as comments, documents, or models are not necessarily consistent or dependable and are cost-prohibitive. To address this situation, we propose the Hybrid Design Pattern Detection (HyDPD), a generalized approach for detecting patterns that is programming-language-agnostic and combines graph analysis (GA) and Machine Learning (ML) to automate the detection of design patterns via source code analysis. Our realization demonstrates its feasibility. An evaluation compared each technique and their combination for three common patterns across a set of 75 single pattern Java and C# public sample pattern projects. The GA component was also used to detect the 23 Gang of Four design patterns across 258 sample C# and Java projects as well as in a large Java project. Performance and scalability were measured. The results show the advantages and potential of a hybrid approach for combining GA with artificial neural networks (ANN) for automated design pattern detection, providing compensating advantages such as reduced false negatives and improved F_1 scores.**

Keywords—software design pattern detection; machine learning; artificial neural networks; graph analysis; software engineering.

I. INTRODUCTION

This paper extends our previous work on automatic design pattern detection (DPD) [1].

A major digitalization transformation is underway throughout industry and society [2], dependent on increasing amounts of software to drive it. For instance, Google is said to have at least 2bn lines of code (LOC) accessed by over 25K developers [3], and GitHub currently reports over 200m repositories and 73m developers [4]. It has been estimated that worldwide well over a trillion LOC exist [5] with 111b lines of new software code created annually [6]. This is exacerbated by a limited supply of programmers and high employee turnover rates for software companies, e.g., 1.1 years at Google [7]. Furthermore, the high degree of utilization in live business operations creates additional time pressure and stress for rapid turnaround, development or maintenance cycles, or deployment times.

The economics of rapidly growing codebases, code longevity, and high turnover make program development and maintenance challenging programmers (herewith including maintainers) especially with regard to fast program comprehension and understanding of (legacy) codebases. Given limited resources and such a vast amount of code, ~75% of technical software workers are estimated to be doing maintenance [8]. Moreover, program comprehension may consume up to 70% of the software engineering effort [9]. Activities involving program comprehension include investigating functionality, internal structures, dependencies, run-time interactions, execution patterns, and program utilization; adding or modifying functionality; assessing the design quality; and domain understanding of the system [10]. Code that is not properly understood by programmers impacts efficiency and reduces quality.

For program comprehension, experts tend to develop efficiently organized specialized schemas or abstractions that contribute to efficient problem and system decomposition and comprehension, and macrostructures (or chunks) and beacons (or cues) being important elements in cognition mental models [11]. In the area of software engineering, software design patterns have been well-documented and popularized, including the Gang of Four (GoF) [12] and POSA [13]. The application of abstracted and documented solutions to recurring software design problems has been a boon to improving software design quality and efficiency. Discovering such common macrostructures or associated pattern terminology in code can serve as beacons to such abstracted macrostructures and may help identify aspects such as the author's intention or purpose.

However, the actual detection and post-coding documentation of these software design patterns remain a challenge. As design patterns have mostly been described informally, their implementation can vary widely, depending on various factors such as the programming language, the natural language and keywords used, the concrete pattern structure, the terminology awareness of the programmer, their experience, and their understanding and (mis)interpretation. Furthermore, the pattern books referenced above were published over 25 years ago and not standardized with regular updates. Pattern variants may occur and patterns may evolve over time with technology. Additionally, the manual documentation of software design patterns usage in project documents such as the architecture specification may not be dependable due to inconsistencies with the codebase, e.g., prescriptive documentation of intentions, adaptations during development, or maintenance changes. Determining actual pattern usage can be beneficial for identifying which patterns

are used where and can help avoid unintended pattern degradation and associated technical debt and quality issues. However, the investment necessary for manual pattern extraction, recovery, and archeology is cost prohibitive and not sustainable due to the high design competency and labor-intensive code analysis effort required, especially in light of the aforementioned codebase sizes and high turnover. Prechelt et al. [14] come to the conclusion that explicit identification of patterns in code (here via manually place pattern comment lines) facilitate faster and less error-prone maintenance tasks.

In source code, patterns may not be explicitly mentioned or commented at all, or they might be or inconsistently or incorrectly mentioned. Semantic issues due to various natural languages and naming differences may also cause beacons or keywords to differ. Since in our context we assume access to source code, intentional obfuscation via a tool is unlikely, but unintentional obfuscation is possible. One way to nevertheless explicitly identify patterns in code would be automated detection via a tool. Yet automated detection and extraction of software design patterns from code is not readily available among popular software development tools. Various research work has attempted to find automated techniques, yet these often fail to recognize or address coverage of all of the basic 23 GoF patterns and rather emphasizing certain design patterns, or were not evaluated on a larger code base. They conclude by suggesting that a combination of techniques might be promising research approach.

In our previous work DPDML (Design Pattern Detection using Machine Learning) [1], we showed the feasibility of our cross-language approach for DPD by realizing the ML core of our approach. Our evaluation using 75 unique Java and C# code projects for training and testing to detect three different types of GoF patterns (creational, structural, and behavioral) provided insights into its potential and limitations. It necessitated finding sufficient training sets (sample projects) for each pattern, and our realization, while combining metrics and semantic analysis, relied on a single technique, namely ML.

This paper contributes our hybrid automated design pattern detection approach called Hybrid Design Pattern Detection (HyDPD) supporting multiple programming languages and amalgamating GA with ML to utilize the advantages of both techniques while decreasing their liabilities. Our realization of the solution approach shows its feasibility. An evaluation compared each technique and their combination for three common patterns across a set of 75 single-pattern Java and C# public sample pattern projects. Furthermore, to provide insights into its potential and limitations, the GA component was applied on 23 GoF design patterns across 258 sample C# and Java projects, as well as a larger Java project (JUnit).

The structure of this paper is as follows: the following section discusses related work. Section 3 describes our solution approach. In Section 4, our realization is presented, which is followed by our evaluation in Section 5. Thereafter, a conclusion is provided.

II. Related Work

Various approaches have been used for software design pattern detection, and they can be categorized based on different analysis styles, such as structural, behavioral, or semantic (and some utilizing a combination of styles) [15]. Structural analysis utilizes static DPD based on inter-class dependencies, data types, and method invocations found in code. Behavioral analysis extracts behavior via static and/or dynamic analysis techniques, since structure alone may not suffice to differentiate patterns. Semantic analysis utilizes naming and annotations to distinguish patterns. Another categorization option is to group by detection methodologies or techniques (learning-based, graph-based, metric-based, etc). As a consequence, some work may fall into multiple categories.

Graph-based approaches include: Yu et al. [16] that reverse engineer code to UML class diagrams and from XMI parse and analyze sub-patterns with class-relationship directed graphs. Mayvan and Rasoolzadegan [17] use a UML semantic graph. Bernardi et al. [18] apply a DSL-driven graph matching approach. DesPaD [19] extract the abstract syntax tree from the code, create a single large graph model of a project, and then apply an isomorphic sub-graph search method using the Subdue tool. Further isomorphic subgraph approaches include Pande et al. [20] and Pradhan et al. [21], both of which begin with UML class diagrams.

Learning-based approaches map the DPD problem to a learning problem, and can involve classification, decision trees, feature maps or vectors, Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), etc. Examples include Alhusain et al. [22], Zanoni et al. [23], Galli et al. [24], Ferenc et al. [25], Uchiyama et al. [26][27], and Dwivedi et al. [28]. Thaller et al. [29] describe a micro-structure-based structural analysis approach based on feature maps. Chihada et al. [30] convert code to class diagrams, which are then transformed to graphs, and have experts create feature vectors for each role based on object-oriented metrics and then apply ML.

Additional approaches include: reasoning-based approaches such as Wang et al. [31] that is based on matrices. Examples of fuzzy logic approaches include Alhusain et al. [32] and Hussain et al. [33]. Examples of rule-based approaches include Sempatrec [34] and FiG [35], which uses an ontology representation. Metric-based approaches include MAPeD [36], PTIDEJ [37], Uchiyama et al. [26][27], and Dwivedi et al. [28]. Fontana et al. [38] analyze microstructures based on an abstract syntax tree. An example semantic-analysis style approach is Issaoui et al. [39]. DP-Miner [40] uses a matrix-based approach based on UML for structural, behavioral, and semantic analysis.

The DPD styles and methodologies used are quite fractured and none has reached a mature and high-quality result with an accessible and executable implementation that we could evaluate. We are not aware of any approach yet that can automatically and reliably detect all 23 GoF design patterns. Most have some limitation or drawback, and the success rate reported among the approaches varies tremendously. Thus, further investigation and research in this

area is essential to enhancing the knowledge surrounding this area. In contrast to the aforementioned work, our HyDPD solution offers a hybrid code-centric approach combining various promising structural, behavioral, and semantic analysis techniques to leverage the strengths of each. In contrast to others, it supports multiple popular programming languages. With HyDPD we show the advantages of combining GA, ML, metrics, and semantic analysis.

III. Solution

Of all available artifacts for DPD, source code represents the reality rather than the intention, and should be readily available, whereas other pattern information (binaries may not build, and runtime instrumentation, UML models, or documentation may not exist). Furthermore, as UML diagrams can be generated by tools from the source code, the underlying data they utilize is also available in the source code. As shown in Figure 1, our solution approach thus begins with source code as the input and is based on the following principles:

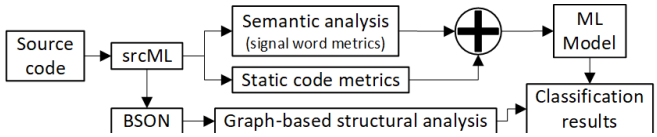


Figure 1. The HyDPD solution concept.

Programming language-independent: since patterns are independent of programming language, our solution abstracts the source code by converting it into an abstracted common format for further processing. For this, our realization currently utilizes srcML [41], which provides an XML-based format, currently supporting C, C++, Java, and C#. If other abstract syntax formats are standardized and available for analysis in a common format, these also can be considered.

Semantic analysis: common pattern signal words in the source code can be used as an indicator or hint for specific pattern usage. Additional natural languages can be supported to detect usage of pattern names or their constituent components in case they were coded in other languages. Our realization supports German, Russian, and French.

Static code metrics extraction: various static code metrics are utilized to detect and differentiate design patterns. The value ranges of metrics are normalized to a scale of 0-1 for utilization with an ANN.

ML model: in utilizing ML to analyze sample data, a model can learn how to classify new unknown data, in our case to differentiate design patterns. Our realization may apply or combine any ML model that suits the situation. Currently, an ANN is used because we were interested in investigating its performance, and intend in future work to detect a wide pattern scope, pattern variants, and new patterns. From our standpoint, alternative non-ML methods such as creating a rule-based system by hand would require labor and expertise as the number of patterns increases and new undiscovered patterns should be detected. With an appropriate ML model, these should be learned automatically and be more readily detected. Challenges include appropriate slicing of the codebase for appropriate metrics and pattern comparison, and finding suitable and large enough training datasets for each pattern.

Graph-based structural analysis: the XML-based code representation is converted to a BSON (Binary JSON) format and stored in a graph database to support graph-based structural analysis. In contrast to ML, the advantages include the ability to apply graph queries across an entire codebase and not requiring any training data. Liabilities include the need for hand-crafted detection queries that are not too specific (thus overlooking many with only slight variations) nor too general or ambiguous to be of practical use (identifying too many false positives).

The underlying hypothesis driving our HyDPD investigation is that amalgamating additional data and metrics in combination with various analysis techniques such as graph and ML models results in better classification accuracy compared to any single technique or metric alone. From a practicality standpoint, our approach could reduce the labor involved in detecting and documenting patterns compared to finding potential patterns manually by perusing code and accurately classifying them (e.g., for assessing or modernizing an unfamiliar legacy system), and can assist developers, maintainers, or experts involved in various software archeology activities.

IV. Realization

The realization of the HyDPD approach consists of two main components: HyDPD-ML that applies the ML technique and HyDPD-GA that applies the GA technique. Python was used to implement the prototype due to its versatility and large selection of available libraries, while a Jupyter Notebook was used for the DPD user interface.

A. HyDPD-ML

Python was used to implement the prototype due to its versatility and the available libraries to support the implementation of ANNs. TensorFlow was chosen along with Keras as a top-layer API.

For ML, a sufficient dataset of different and realistic projects was needed to support supervised learning. While certain pattern examples in code can readily be found, finding a larger set of different ones in different project settings and programming languages turns out to encounter various practical challenges and is labor intensive. Due to resource and time constraints, our ML realization thus initially focused on having the network learn to detect one pattern out of each of three main pattern categories: from the structural category - Adapter; from the creational patterns -Factory; and from the behavioral patterns - Observer. Future work will expand the pattern scope.

Metric-based matching: The ElementTree parser was used to traverse srcML and count the specific XML-tags. The metric values were not separated by roles or classes, but are merged and evaluated as a whole. The metrics used were inspired by Uchiyama et al. [26] and are shown in Table I.

TABLE I. OVERVIEW OF METRICS

Abbreviation	Description
NOC	Number of classes
NOF	Number of fields
NOSF	Number of static fields
NOM	Number of methods
NOSM	Number of static methods
NOI	Number of interfaces
NOAI	Number of abstract interfaces

TABLE II. SIGNAL WORDS FOR DESIGN PATTERNS

Pattern	Signal Words			
Adapter	Adapter	adaptee	target	adapt
Factory	Factory	create	implements	type
Observer	observer	state	update	notify

Semantic-based matching: An obvious approach to pattern detection is naming. If a developer already used common design pattern terminology in the code, then this should be utilized as a pattern detection indicator. For our signal word detection, we translated the signal words to German, French, and Russian to improve results for non-English code.

Semantic variations: To determine if other signal words beyond the design pattern name were used in implementations, we analyzed several examples of implemented design patterns. 12 additional signal words were selected, four for each pattern as shown in Table II.

Internationalization: To test internationalization, the Python library *translate* was used to translate the signal words to German, French, and Russian. Rather than extending the list of metrics passed to the ANN, a match with a translated word is counted in the same input parameter as the original English words. Applying Natural Language Processing (NLP) to reduce words by stemming or creating lemmas to compare to a defined word list would also be possible, and may improve or deteriorate the results, if for instance the input array contained further zeros when no signal words were found.

ANN: Based on our realization scope, since the input array is not multidimensional, deep neural networks (DNNs) with additional layers would not necessarily yield improved results. We thus chose to realize an Artificial Neural Network (ANN) with one input layer, two hidden layers, and one output layer as shown in Figure 2. We created the network with the Keras API with the TensorFlow Python library.

The input layer size matches the data points, and as there are 7 metrics and 12 semantic match values, this makes 19 input values total. The input model structure is a numpy array as follows:

```
[NOC, NOF, NOSF, NOM, NOSM, NOI, NOAI, ASW1,
  ASW2, ASW3, ASW4, FSW1, FSW2, FSW3, FSW4,
          OSW1, OSW2, OSW3, OSW4]
```

The first 7 values correspond to Table I while the rest indicate the number of signal word matches from Table II. SW=Signal Word, A=Adapter, F=Façade, and O=Observer, 1-4 implies the corresponding table column. Only 7 metric values are utilized when no signal words exist.

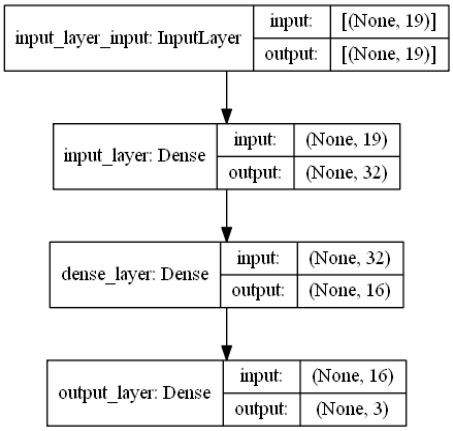


Figure 2. ANN model overview created with Keras.

The first hidden layer is a dense layer (with each neuron fully connected to the neurons in the prior layer) consisting of 32 neurons. The activation function was a rectified linear unit (ReLU). The second layer is a dense layer with 16 neurons. This conforms with the general guideline to gradually decrease the neurons as one approaches the output layer. The output layer consists of three neurons to match the three design patterns that should be detected. The "Softmax" activation method is used, which is often used in classification problems and supports identifying the confidence of the network in its decision. The "Adam" algorithm is a universal optimizer that is recommended in a wide assortment of papers and guides. As no specialized optimizer was needed, "Adam" with its default values was chosen as defined in [42]. No regularization was applied in each layer. Adam automatically adjusts and optimizes the learning rate. Sparse categorical cross entropy was applied as the loss function for this multi-class classification task.

The size of the ANN should fit the size of the problem. Small adjustments to the ANN structure showed no significant performance impact, whereas significantly increasing the neuron count or layer count negatively impacted results. With two hidden layers and 48 neurons, the first layer contains 640 parameters, the second layer 528, and the output layer 51, resulting in 1219 parameters that are adjusted during training.

The network is trained in epochs, wherein the complete training set is sent through the network with weights adjusted. As the weights and metrics change per epoch, an early-stopping callback stops the training if the accuracy of the network decreases over more than 10 epochs, saving the network that had the best accuracy. A validation dataset is typically used during training to monitor results on unlearned data after each epoch, but as our training set was limited, we used a prepared testing dataset with known labels.

1) *Training Datasets*

As to possible design pattern training sets, the Pattern-like Micro-Architecture Repository (P-MARt) includes a collection of microstructures found in different repositories such as Jhotdraw and JUnit. However, because these patterns are intertwined with each other, they do not provide isolated example specimens for training the ANN. The Perceptrons

Reuse Repositories could theoretically provide many instances of design patterns for a training dataset, but no results were provided on the website during the timeframe of our realization, and while the source code analyzer is free, the servers could not be reached.

We did manage to find training data as detailed in the next section. Since our initial intent for HyDPD was a much broader scope for data pattern mining, and because we expected a large supply of sample data, we focused on an ANN realization. We were also interested in determining if we could train an ANN to detect these patterns with relatively few samples. However, due to unexpected additional resource and time constraints involved in finding pattern samples manually, we had to reduce the number of design patterns involved, and could not compare the ANN with alternative classification schemes such as Naïve Bayes, Decision Tree, Logistic Regression, and SVMs, but intend to in future work.

B. HyDPD-GA

For the GA realization, the srcML is converted to BSON and stored in MongoDB. A Neo4j Cypher procedure is then used to import the BSON from MongoDB into the Neo4j database. `OPTIONAL MATCH` is used to permit variations of patterns to be in the result set, while missing entities are notated with `None`. The result set is provided as a `Dataframe` structure. Figure 3 shows the Python classes used for the implementation. Not all methods in the Python project are depicted in the diagram. Some methods have been defined in the scope of view of the class methods. They are only used within the framework of the respective method and serve to improve the readability of the code. For example, the method `resolve_names` in the `ResolveJsonTypes` class contains another 24 methods.

The `FileStorage` class is required to create the folders in advance in which the intermediate results of program execution are stored. All classes other than `DPDetector` use the `Settings` class to query the values of the current configuration parameters (e.B. Uniform Resource Locator (URL) of the MongoDB). The `SrcmlDriver` class is used to convert the code project into the srcML XML representation. The `SrcmlJsonConverterMulti` class is used to detect whether the specified code project was written in Java or C#. Depending on this, either the class `SrcmlJsonConverterForCSharp` or `SrcmlJsonConverterForJava` is used to convert the XML representation of the project to JSON. Both classes inherit from the abstract class `SrcmlJsonConverter`, which contains shared functionality. In addition, the `SrcmlJsonConverter` class interacts with the `ResolveJsonTypes`, which undertakes part of the conversion process. In addition, all classes (except `ResolveJsonTypes`) use the `SrcmlUtil` class, which supports parsing the XML representation of the project. The `TypesNeo` class interacts with the `TypesMongo` class to import a specific project from MongoDB into the Neo4j database. The `DPDetector` class uses the `TypesNeo` class when executing the DPD queries on the Neo4j database.

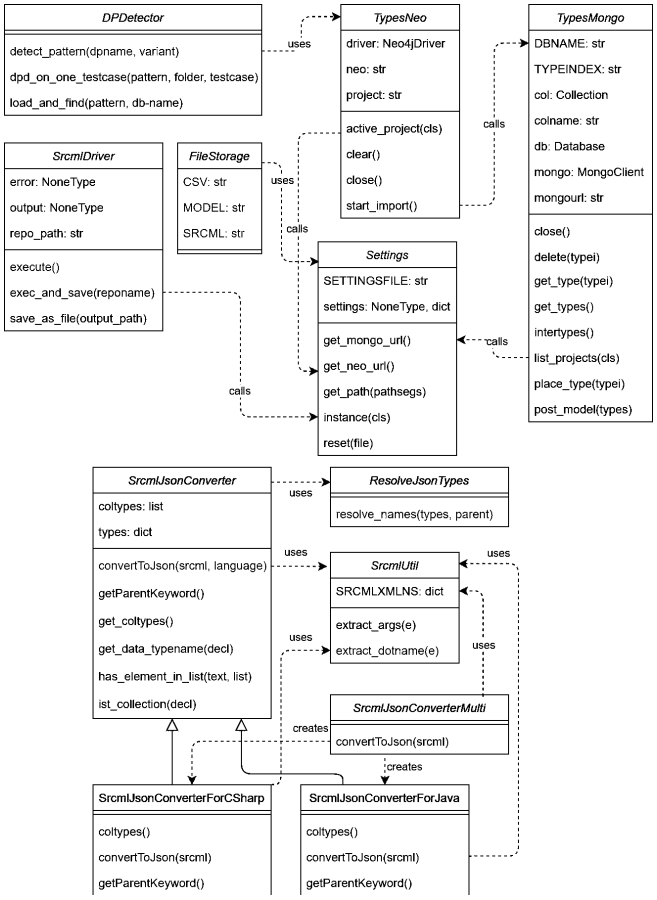


Figure 3. HyDPD-GA Python classes (partial view).

For each parsed class, the full class name, available imported entities, and inheritance from other classes and interfaces are extracted. A flag indicates whether the respective class is abstract. For each attribute, the type, name, and two flags are used for indicating a static attribute or if it involves a collection. The parsed class contains a collection of methods with their implementation, abstract methods, constructors, and getter/setter methods taken into account. For each method, the following data is extracted:

- method name;
- static flag;
- abstract flag;
- constructor flag;
- return type;
- flag whether the method is accessible outside its class;
- method arguments;
- declared variables including name, type, and collection flag;
- type assigned when a variable was initialized; and
- names of other methods called from this method.

After all relevant information has been extracted and stored in the form of the JSON object in the `types` attribute, the second stage of processing takes place. This consists in resolving the name via all possible types and methods. For this, the static class `ResolveJsonTypes` or its method `resolve_names` is utilized. All possible types refers to the

inherited classes, class attributes, method variables, method arguments, and the type a method returns. Resolving methods considers the methods that are called as well as the methods that are used for initialization, and is based on the imported entities, the attributes and methods of existing and inherited classes, and the local variables. Moreover, the relationship of overriding a parent method via a method in the class being analyzed is included. In addition, the attributes and methods of the parent class are added to the inheriting class.

The method is then executed `transform_types` to convert the obtained JSON format to the mongoDB BSON format. The result is returned by the `convertToJson` method in the `SrcmlJsonConterverMulti` class. To store the BSON object in MongoDB, a new object of the `TypesMongo` class is instantiated. It then calls the method `transformed_types` and passes the BSON object as input to that method. Finally, the code project is saved in BSON format in MongoDB.

The class `DPDetector` performs pattern recognition. The method `dpd_on_one_testcase` in this class removes any preexisting Neo4j entities and searches for a single pattern in the specified project. It instantiates a new object of the class `TypesNeo`, importing the corresponding collection from MongoDB into the Neo4j database using a Cypher query. The `detect_pattern` method uses a Cypher query to search for a designated design pattern. If matches with the query exist, the result records are grouped by the main participant of the pattern, and the correctness of the match is calculated as a proportion of the matching nodes relative to the total number of nodes searched for (e.g., 0.70 would indicate a 70% query match). The resulting records are then returned as a `DataFrame`.

An example Cypher query for the Chain of Responsibility (CoR) pattern is shown in Figure 4. First, the type `handler` is defined that has two methods: `handle_op` and `set_op`. The `set_op` has at least one argument of type `handler`. Furthermore, two types `c_handler` and `c_handler2` that inherit from the `handler` are defined. In addition, both classes must have a `handler` attribute and a method that overrides the parent class's method.

There should also be a `client` type with a method `client_op`. The `client_op` method should call either `handle_op` or `c_handle_op`. In the following, negative conditions are defined that further refine the query and distinguish it from other patterns. The `client` type cannot inherit from the `handler` type. The type `clientc_handler` and `c_handler2` must not have a collection of the type `handler`.

The Cypher query for each pattern was tuned as follows: For each pattern, a test dataset of at least six Java and C# examples from an internet search of GitHub and other sources was used to tune the Cypher query in such a way that it detects the expected pattern (true positives or TPs). In case it did not, the code was analyzed to determine the underlying cause and, if possible, the query or, if appropriate due to a faulty implementation of the pattern, the code adjusted until it is detected. If not, the underlying cause was identified; reasons included for instance 1) a non-compliant application of the pattern that violate core structural rules of the pattern (intentionally or unintentionally due to an author's misunderstanding), 2) referencing external classes (missing) that were not contained in the source code and required to fulfill the pattern, 3) the use of functional programming constructs.

```
MATCH
(handler:Type:Abstract)-[:HAS]->(handle_op:
  Operation),
(handler)-[:HAS]->(set_op:Operation),
(set_op)-[:HAS_ARG]->(handler),

(c_handler:Type)-[:INHERITS*1..3]->(handler),
(c_handler)-[:HAS]->(handler),
(c_handler)-[:HAS]->(c_handle_op:Operation),
(c_handle_op)-[:OVERRIDES*1..3]->(handle_op),

(c_handler2:Type)-[:INHERITS*1..3]->(handler),
(c_handler2)-[:HAS]->(handler),
(c_handler2)-[:HAS]->(c_handle_op2:Operation),
(c_handle_op2)-[:OVERRIDES*1..3]->(handle_op),

(client:Type)-[:HAS]->(client_op:Operation)

WHERE ((client_op)-[:CALLS]->(handle_op)
OR (client_op)-[:CALLS]->(c_handle_op))
AND NOT (client)-[:INHERITS]->(handler)
AND NOT (c_handler)-[:HAS {collection: true}]->(
  handler)
AND NOT (c_handler2)-[:HAS {collection: true}]->(
  handler)

RETURN *
```

Figure 4. Cypher query for the Chain of Responsibility pattern.

Thereafter, each query was executed on all 22 other patterns to determine what should not belong to the query result with regard to FPs. If a positive in an unexpected pattern was detected, it was analyzed to determine if 1) the query must be further tuned, 2) something in the target code example is inappropriate (abstract method, a participant is not allowed in an inheritance hierarchy, optional pattern elements), etc.

An example excerpt from the result output of a Jupyter Notebook is shown in Figure 5.

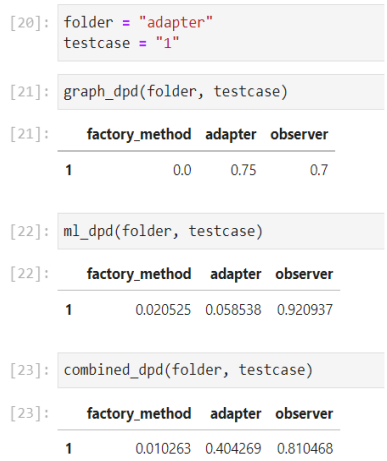


Figure 5. Example Jupyter Notebook result output excerpt.

C. HyDPD

To enhance DPD, the HyDPD realization combines both HyDPD-ML and HyDPD-GA components, with each component supplying a probability *P* as shown in (1) about the likelihood of a certain pattern being detected, where *w* is the weighting. *w* is currently equal and set to 0.5 until further empirical insights are gathered as to which approach is typically more accurate.

$$P = \omega_{ML} * P_{ML} + \omega_{GA} * P_{GA} \quad (1)$$

Weight Tailoring: The weightings can be tailored across all patterns or on a per pattern basis, for instance based on empirical accuracy rates if one technique is determined to have better accuracy for a specific design pattern.

V. EVALUATION

Since the primary contribution in this paper is the new GA component and its hybrid combination with ML, this evaluation focused primarily on investigating the potential of GA and its utilization in combination with ML. Our research questions (**RQ**s) are adjusted to the limitations of our available datasets and time and resources.

The first three RQs utilize three common GoF patterns for analyzing and comparing the HyDPD components and the hybrid approach, since HyDPD-ML requires larger pattern-specific datasets for training:

RQ1. *How does HyDPD-ML perform against three common GoF patterns?*

RQ2. *How does HyDPD-GA perform against three common GoF patterns, and how does it compare with HyDPD-ML?*

RQ3. *How does the hybrid HyDPD perform against three common GoF patterns, and how does it compare with HyDPD-ML and HyDPD-GA?*

The next four RQs focus on analyzing HyDPD-GA's capabilities:

RQ4. *Can HyDPD-GA detect more abstract architectural patterns, in particular the Model-View-Controller (MVC) pattern?*

RQ5. *How does HyDPD-GA perform against the 23 GoF patterns and in comparison to related work?*

RQ6. *How does HyDPD-GA and HyDPD-ML perform against a large project in comparison to related work?*

RQ7. *What performance latency and scalability can one expect with using HyDPD-GA and how does it compare with HyDPD-ML?*

The evaluation consisted of seven parts, each addressing a research question: A) HyDPD-ML with three common design patterns, B) HyDPD-GA with three common design patterns, C) hybrid HyDPD with the same patterns, D) HyDPD-GA to probe its ability with an architecture pattern: MVC, E) HyDPD-GA across all 23 GoF patterns, and F) HyDPD-GA performance latency and scalability

The software configuration used in the evaluation consisted of srcML v1.0, Python 3.8, MongoDB v4, and Neo4j 4.2. Python libraries included: simplejson 3.17.4, pymongo 3.12.0, neo4j 4.2.0, tensorflow 2.6.0, googletrans 3.1.0a0, scikit-learn 0.24.2, keras 2.6.0, beautifulsoup4 4.9.3, numpy 1.19.5, matplotlib 3.4.3, conda 4.10.3, dpd 0.0.1, dpdml 0.0.1, scipy 1.7.1, pip 21.2.4, pandas 1.3.2, jupyter-client 6.1.12, jupyter-core 4.7.1, jupyter-server 1.10.2. The hardware configuration consisted of a PC with an i5-10210U@1.6GHz CPU, 8GB RAM, 1TB SSD running W10 Home. Docker v20.10.8 was used to containerize the services: jupyter-notebook, neo4j, mongo, and mongo-gui.

Dirty datasets: While it would be feasible to only involve both pure training and pure test datasets (manually removing or fixing all incorrectly implemented or mislabeled projects) and thus boost the accuracy numbers, we instead chose to include the real-world datasets as they were labeled by the authors of the projects we found, even though the authors may have incorrectly implemented or labeled the patterns. Thus, we intentionally include real-world impurity and our calculated accuracies reflect this, rather than achieving the 100% accuracy possible had we manually precleaned training and test datasets.

A. HyDPD-ML Evaluation (Three Patterns)

To address **RQ1** and serve as a basis for comparison, the HyDPD-ML evaluation consisted of three steps: 1) dataset acquisition, 2) supervised training, and 3) testing.

1) Dataset Acquisition

It remains a challenge to procure sufficient code projects with implemented pattern datasets in different programming languages and various patterns for both training and testing an ANN. Due to resource constraints, we thus focused on three common patterns from each of the major pattern categories: from the creational patterns, Factory; from the structural category, Adapter; and from the behavioral patterns, Observer. We then found 25 unique single-pattern code projects per pattern small single-pattern code projects from public repositories, 49 in Java and 26 in C# (mostly from github and the rest from pattern book sites, MSDN, etc.), evenly distributed into as shown in Figure 6. They were specifically labeled as examples of these patterns and manually verified. These popular programming languages are supported by srcML, and the mix of languages permits us to demonstrate the programming language independent principle. Language inequalities between available pattern examples is likely attributable to the popularity and longevity of a language and interest in patterns in that community.

Training data: Applying hold-out validation, of the 75 projects available, we selected 60 (20 per pattern category) for training the ANN, with between 60-75% of the code projects being in Java (green) and the remainder in C# (blue) as shown in the upper section of Figure 6.

Test data: The remaining 15 projects of the 75 total (five per pattern category with three in Java and two in C#) were used for the test dataset. In order to test whether signal word pattern matching significantly impacts the ANN results, these projects were duplicated and their signal words removed or renamed, resulting in six Java (orange) and four C# (purple) projects per pattern/category as shown in the lower section of Figure 6. This resulted in 10 test projects per pattern or 30 total test projects.

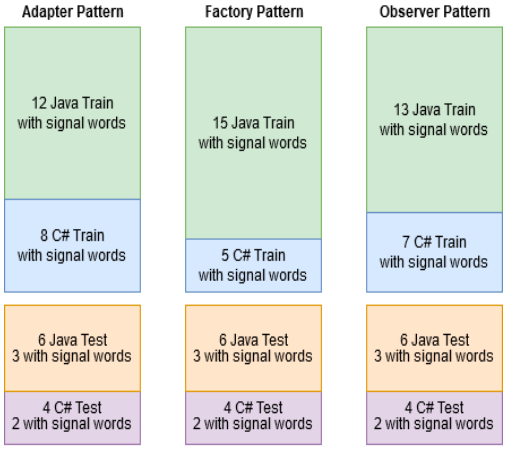


Figure 6. Pattern-specific datasets in columns with programming language specific training sets on the top rows and test sets on the bottom.

2) Supervised Training

As shown in Figure 7, during training the accuracy improves from 47% to 95% in the first seven epochs, thereafter fluctuating between 85-95% with a peak of 96.7% in the 27th epoch. The network loss metrics are shown in Figure 7. The loss value drops from an initial 1.0841 to 0.2816 in epoch 17 before small fluctuations begin, with the trend continuing downward. The loss value of 0.1995 in epoch 27 is an adequate prerequisite for detecting patterns in unknown code projects, and we saw little value in increasing the training epochs. The early stopping callback was not triggered since the overall accuracy of the network is still increasing despite the fluctuations, indicating a positive learning behavior and implying that with the given data points, it is finding structures and values that allow it to differentiate the three design patterns from each other. We thus chose to stop the training at 30 epochs, which took 2-45 seconds depending on the underlying hardware environment (any Graphical Processing Unit (GPU) with CUDA support will improve processing times).

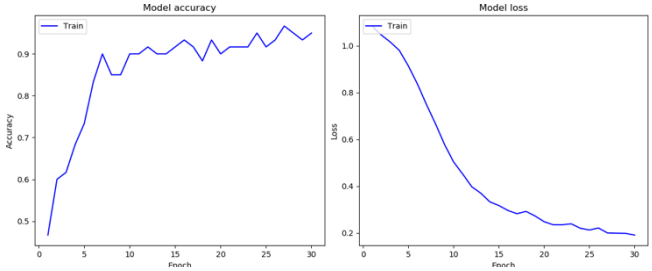


Figure 7. Network accuracy and loss over 30 epochs of training.

Considering that the worst case of random guessing would result in an accuracy of 33%, the accuracy result of 97% is significantly better and shows the potential of the approach.

The training results show that not only is the ANN learning to differentiate the patterns, its confidence for these determinations increases during the training. By epoch 27 with an accuracy of 96.7% and a loss of 0.1995, only two out of the 60 total training projects spread evenly across the three design patterns are incorrectly classified.

3) Testing

For the test dataset, 15 unique code projects were selected (five unique projects per pattern), and these were then duplicated and their signal words removed, resulting in 30 code projects. By removing the signal words, we can determine the degree of dependence of the network on these signal words.

During testing, the reported accuracy dropped to 83.3%, meaning 25 of the 30 patterns were correctly identified. Furthermore, the loss went to 0.4060, meaning a loss in confidence of its determination. A deterioration in these values is to be expected when working with unfamiliar data.

The resulting confusion matrix is shown in Table III, showing that the network was able to use its learned knowledge in training to correctly classify a majority of unknown projects (25 out of the 30 test projects). The precision column indicates how many of the predicted labels are correct, while the recall row indicates how many true labels were predicted correctly. Fewer false positives (FPs) improve the precision, while fewer false negatives (FNs) improve the recall value. All the code projects predicted to be Factory were correct (a precision of 100%), while the remaining 30% of the Factory pattern projects were incorrectly classified as another pattern, resulting in a recall of 0.70. This indicates that the Factory is more easily confused with the other patterns, a possible explanation being that the metrics we used may better differentiate more involved (i.e., more complex) patterns. The other patterns had less precision (0.81 or 0.75), but a better recall of 0.90. The overall accuracy is 88.9% with an F_1 score of 0.83. In one Observer testcase, HyDPD-ML was evenly split with Factory Method (0.46 vs. 0.46) and thus categorized as a FP.

As to the influence of signal words, our hypothesis that signal words would improve the results proved hitherto unfounded. The classification precision was not affected by signal words, with 12 projects with signal words and 13 without being correctly classified. Additional test runs showed similar results (+/- one project).

TABLE III. CONFUSION MATRIX: ML TEST 10 PROJECTS PER PATTERN

Predicted Labels	**True Labels**			**Accur.**	**Precision**	**Recall**	**F_1 Score**
	Factory	***Adapter***	***Observer***				
Factory	7	0	0	90.0%	1.00	0.70	0.82
Adapter	1	9	1	90.0%	0.81	0.90	0.86
Observer	2	1	9	86.7%	0.75	0.90	0.82
Overall				88.9%	0.83	0.83	0.83

Accur. = Accuracy

TABLE IV. CONFUSION MATRIX: ML CROSS-TESTING 90 PROJECTS

Predicted Labels	**True Labels**			**Accur.**	**Precision**	**Recall**	**F_1 Score**
	Factory	***Adapter***	***Observer***				
Factory	24	0	1	92.2%	0.96	0.80	0.87
Adapter	0	24	0	93.3%	1.00	0.80	0.89
Observer	6	6	29	85.6%	0.71	0.97	0.82
Overall				90.7%	0.87	0.86	0.86

Accur. = Accuracy

Since HyDPD-GA requires no training set, and we will be comparing HyDPD-ML with HyDPD-GA and the combination as HyDPD, it is pragmatic to utilize the entire

dataset of 90 projects for our testing. Hence, we also applied ML across the entire dataset (including the training set) to serve as a reference for comparison and to ensure that DPD did not get worse when the training set is also used for testing. This result, which includes the training dataset, is shown in Table IV, showing that overall accuracy increased to 90.3%, with precision and recall increasing to 0.86 with an overall F_1 score of 0.86.

B. HyDPD-GA Evaluation (Three GoF patterns)

To answer **RQ2** and to be able to compare HyDPD-GA with HyDPD-ML for the three GoF patterns, we utilized the entire ML dataset (both the training and test data) as the test dataset at 30 test projects per pattern and 90 total. The results are shown in Table V-VII below, where for this section we are focused on the columns HyDPD-GA and its comparison to HyDPD-ML (the column HyDPD will be discussed in the following section). The Testcase ID is unique only within a specific pattern (for internal tracking), so the ID may reoccur within another pattern and refers to a different test case.

TABLE V. DPD COMPARISON: FACTORY PATTERN

Testcase	HyDPD-ML	HyDPD-GA	HyDPD
1	1.00	**0.00**	0.50
10	1.00	0.70	0.86
11	1.00	0.70	0.86
12	1.00	1.00	1.00
13	0.98	0.70	0.85
14cs	1.00	1.00	1.00
15cs	1.00	1.00	1.00
16cs	1.00	1.00	1.00
17cs	0.99	**0.00**	**0.49**
18	1.00	1.00	1.00
19cs	1.00	1.00	1.00
2	1.00	1.00	1.00
20	0.99	1.00	1.00
21	1.00	0.70	0.86
22	**0.04**	1.00	0.52
23	0.98	1.00	0.99
24cs	0.99	1.00	0.99
25cs	1.00	1.00	1.00
26	**0.02**	0.70	**0.37**
27	**0.03**	1.00	0.51
28	**0.02**	1.00	0.51
29cs	**0.03**	1.00	0.51
3	1.00	0.70	0.86
30cs	**0.05**	1.00	0.53
4	0.99	0.70	0.85
5	1.00	0.70	0.86
6	0.67	0.70	0.69
7	1.00	0.70	0.86
8	0.86	0.70	0.78
9	0.99	0.70	0.85
FN*	6	2	2

*False Negatives marked in **bold** above

In applying GA to the test datasets, certain testcases returned less than the ideal value of 1.0 (e.g., 0.75 would indicate a partial match and 0 no match). Since GA works differently than ML and can identify a specific node involved in a pattern, we can utilize the results to analyze the cause. A manual analysis found the following explanations for the discrepancies (non 1.0 values) in the HyDPD-GA column in Tables V-VII:

TABLE VI. DPD COMPARISON: ADAPTER PATTERN

Testcase	HyDPD-ML	HyDPD-GA	HyDPD
1	**0.06**	0.75	**0.40**
10	1.00	1.00	0.87
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13cs	1.00	1.00	1.00
14cs	1.00	1.00	1.00
15cs	1.00	0.75	0.87
16cs	1.00	1.00	1.00
17cs	1.00	1.00	1.00
18cs	1.00	0.75	0.87
19cs	1.00	1.00	1.00
2	1.00	**0.00**	0.50
20cs	1.00	1.00	1.00
21cs	1.00	1.00	1.00
22cs	0.89	1.00	0.95
23	1.00	1.00	1.00
24	1.00	1.00	1.00
25	0.71	1.00	0.86
26cs	**0.07**	1.00	0.54
27cs	**0.04**	1.00	0.52
28	**0.05**	1.00	0.52
29	**0.06**	1.00	0.53
3	1.00	1.00	1.00
30	**0.04**	1.00	0.52
4	0.64	**0.00**	**0.32**
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	**0.00**	0.50
8	1.00	1.00	1.00
9	1.00	1.00	1.00
FN*	6	3	2

* False Negatives marked in **bold** above

Factory pattern: 1) missing either an abstract Factory, an abstract Factory Method, or both, 2) using Builder and functional programming, 3) Factory Method does not return a Product (offers a getter call to retrieve it).

Adapter pattern: 1) client missing or call of target method missing, i.e., code just shows a possible implementation without invoking the pattern 2) Adaptee calls the Adapter's method – but an Adaptee should not have to know about the adaptation (GA returns 0) 3) two different methods are used, one method overrides the target method and another method calls the Adaptee's method, 4) the Adapter does not actually call the Adaptee (GA returns 0).

Observer pattern: 1) missing abstract Subject or abstract methods for notification, 2) missing methods to add or remove Observers from the internal list, 3) an external iterator, events, or delegation was used (GA returns 0).

The results show no overlap of FNs occurred across all 90 test cases - an indication of how each component differs and how they can be used together to complement one another. HyDPD-GA performed as good or better than HyDPD-ML for each pattern with the exception of the Observer pattern with 2 FNs (False Negatives) versus 1 FN for HyDPD-ML.

Table VIII shows the result for cross-testing the three patterns across the 90 test cases resulting in 270 tests in total (TN = true negative). HyDPD-GA had a total of 7 FNs and 1

FP across the testcases, which compares well against the 13 FNs and 13 FPs for HyDPD-ML. Overall HyDPD-GA shows as good or better recall, precision, accuracy, and F_1 scores than HyDPD-ML, with the exception of the single FP for the Adapter pattern resulting in 96.4% vs. 100% precision, and the additional FN in the Observer pattern resulting in a recall of 93.3% vs. 96.7%.

TABLE VII. DPD COMPARISON: OBSERVER PATTERN

Testcase	HyDPD-ML	HyDPD-GA	HyDPD
1	1.00	0.70	0.85
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13cs	0.99	**0.00**	**0.49**
14cs	1.00	1.00	1.00
15cs	1.00	1.00	1.00
16cs	1.00	1.00	1.00
17	1.00	0.70	0.85
18cs	1.00	**0.00**	0.50
19cs	1.00	1.00	1.00
2	1.00	1.00	1.00
20cs	1.00	1.00	1.00
21cs	1.00	1.00	1.00
22cs	0.95	1.00	0.97
23	1.00	1.00	1.00
24	0.95	1.00	0.97
25	1.00	1.00	1.00
26cs	0.91	1.00	0.95
27cs	0.95	1.00	0.97
28	0.94	1.00	0.97
29	0.95	1.00	0.97
3	**0.46**	0.70	0.58
30	0.94	1.00	0.97
4	1.00	1.00	1.00
5	1.00	0.70	0.85
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	0.70	0.85
FN*	1	2	1

* False Negatives marked in **bold** above

C. HyDPD Evaluation (Three GoF Patterns)

To answer **RQ3**, for all three patterns corresponding to Table V through Table VII, the hybrid probability (1) was calculated from the HyDPD-ML and HyDPD-GA results, with the result shown in column HyDPD. Across all three patterns, every single FN from either of the techniques was compensated by a partial or full detection by the other technique, with 0.32 for Adapter testcase 4 being the lowest combined score. the combination often compensates for a FN from another component as can be seen in the tables with the bold FNs. Furthermore, in practical use perhaps a threshold such as 0.3 instead of 0.5 could be used to trigger detection.

Thus, the resulting combination as HyDPD provides a recall as good or better than any single component. Thus, HyDPD improves DPD by compensating for a FN of any isolated HyDPD-ML or HyDPD-GA value, since one may not detect a pattern that the other component can. While this may result in more FPs, we are of the opinion that the benefits of automation improve efficiency sufficiently that one would rather manually quickly verify a detection as false (FP) rather than misleading FNs. Thus, we prefer to minimize the miss rate or false negative rate (FNR).

TABLE VIII. 270 CROSS-TEST DPD SUMMARY

Component	Result	Factory	Adapter	Observer	Total
HyDPD-ML	FP	0	0	12	13
	FN	6	6	1	13
	TP	24	24	29	77
	TN	60	60	48	167
	Recall	80.0%	80.0%	96.7%	85.6%
	Precision	100.0%	100.0%	70.7%	85.6%
	Accuracy	93.3%	93.3%	85.6%	90.4%
	F_1 Score	88.9%	88.9%	81.7%	85.6%
HyDPD-GA	FP	0	1	0	1
	FN	2	3	2	7
	TP	28	27	28	83
	TN	60	59	60	179
	Recall	93.3%	90.0%	93.3%	92.2%
	Precision	100.0%	96.4%	100.0%	98.8%
	Accuracy	97.8%	95.6%	97.8%	97.0%
	F_1 Score	96.6%	93.1%	96.6%	95.4%
HyDPD	FP	0	1	0	1
	FN	2	2	1	5
	TP	28	28	29	85
	TN	60	59	60	179
	Recall	93.3%	93.3%	96.7%	94.4%
	Precision	100.0%	96.6%	100.0%	98.8%
	Accuracy	97.8%	96.7%	98.9%	97.8%
	F_1 Score	96.6%	94.9%	98.3%	96.6%

HyDPD results in Table VIII show improved results, with only 5 FNs and 1 FP out of the 270 test cases, resulting in 94.4% recall, 98.8% precision, 97.8% accuracy, and an F1 score of 96.6%. In all cases, HyDPD provided as good or better results than either HyDPD-ML or HyDPD-GA alone.

D. HyDPD-GA Evaluation (MVC Architectural Pattern)

With regard to **RQ4**, 20 MVC test pattern examples (16 in Java and 4 in C#) were acquired and HyDPD-GA applied. The results were 16 TPs, 4 FNs, and 10 FPs. Recall is 0.80, precision 0.62, accuracy 0.53, with an F_1 score of 0.70. The FNs were due to alternative implementations that deviated from the formal pattern expectation, while the high number of FPs was due to keeping the Cypher query abstract in order to maximize DPD given the numerous possibilities the architectural pattern could be implemented. We are of the opinion that we would rather verify a positive and determine a FP than miss a DPD due to a FN. Thus, we prefer to minimize the miss rate or FNR. Despite the worse results in comparison to the three GoF patterns, we believe HyDPD-GA to be a potentially promising technique for architectural pattern detection as well, and intend to investigate this further in future work.

E. Evaluation of HyDPD-GA with all GoF Patterns

RQ5 focuses on the 23 GoF patterns. Due to resource and time constraints, it was not feasible to train and evaluate the HyDPD-ML component (both alone and in conjunction with GA) against all remaining 20 GoF patterns at 25 sample projects per pattern, which would require the manual acquisition of an additional 500 code project samples. As

HyDPD-GA performed well with relatively good accuracy for the three patterns evaluated previously in sections B and C, and since it requires no training sets, our GoF evaluation only utilized HyDPD-GA. For the remaining 20 GoF patterns, from GitHub and further sources we acquired at least 6 pattern examples (3 in Java und 3 in C#) per GoF design pattern as a test dataset.

1) Testdata

The Cypher query for each pattern was applied to its own pattern test data and tuned as described in the previous section IV.B to maximize its TP and TN. Then the queries were applied to the entire GoF pattern test set consisting of 258 tests. The cross-testing resulted in 5934 tests being executed. A result of 1 was treated as positive, 0 negative, and in-between values manually analyzed. Table IX shows the GoF DPD results, where X stands for Cross-pattern detection and indicates the number of unexpected detections of that pattern in a different pattern test set. These deviations were then manually analyzed to determine if that pattern did indeed occur in the other test set or if it was a FP, shown in the corresponding column.

TABLE IX. HYDPD-GA GOF DPD.

	TC	FN	X	FP	TP	TN	A	P	R	F_1
Abstract Factory	6	0	1	0	7	251	1.00	1.00	1.00	1.00
Builder	9	0	2	2	9	247	0.99	0.82	1.00	0.90
Factory Method	40	2	26	2	62	192	0.98	0.97	0.97	0.97
Prototype	12	1	0	0	11	246	1.00	1.00	0.92	0.96
Singleton	8	0	0	0	8	250	1.00	1.00	1.00	1.00
Adapter	33	3	15	15	30	210	0.93	0.67	0.91	0.77
Bridge	7	0	1	0	8	250	1.00	1.00	1.00	1.00
Composite	10	0	10	0	20	238	1.00	1.00	1.00	1.00
Decorator	14	0	6	6	14	238	0.98	0.70	1.00	0.82
Façade	6	1	1	1	5	251	0.99	0.83	0.83	0.83
Flyweight	14	1	0	0	13	244	1.00	1.00	0.93	0.96
Proxy	6	1	6	6	5	246	0.97	0.45	0.83	0.59
CoR	6	0	0	0	6	252	1.00	1.00	1.00	1.00
Command	6	0	1	0	7	251	1.00	1.00	1.00	1.00
Interpreter	6	1	3	3	5	249	0.98	0.63	0.83	0.71
Iterator	6	1	0	0	5	252	1.00	1.00	0.83	0.91
Mediator	6	1	0	0	5	252	1.00	1.00	0.83	0.91
Memento	6	0	0	0	6	252	1.00	1.00	1.00	1.00
Observer	30	3	3	3	27	225	0.98	0.90	0.90	0.90
State	7	0	5	5	7	246	0.98	0.58	1.00	0.74
Strategy	6	1	6	3	8	246	0.98	0.73	0.89	0.80
Template Method	7	0	5	0	12	246	1.00	1.00	1.00	1.00
Visitor	7	1	0	0	6	251	1.00	1.00	0.86	0.92
Total	258	17	91	46	286	5585	0.99	0.86	0.94	0.90

X = Cross-pattern detection; A=Accuracy; P=Precision; R=Recall

A brief explanation of the FNs and FPs in Table IX:

Builder: FPs were detected in Memento, whereby an Originator instantiates the Memento object based on its own state, resulting in similar behavior.

Factory, *Adapter*, and *Observer*: the FNs are described above in Section B. One FP each in Composite and Façade.

Prototype: FN: a clone method calls a Dictionary object, resulting in an incomplete graph mapping.

Decorator: FPs: the DPD confusion occurs since Decorator, Adapter, Proxy, Interpreter, and State have structural similarities and primarily behavioral differences or differences of intent. Also, the main participant inherits functionalities from an abstract interface and has a reference to an object with this interface.

Façade: FN: the Cypher query required that the Façade class use at least 3 independent classes, but the test case uses an inheritance hierarchy, an atypical realization of the pattern.

Flyweight: FN: missing abstract Flyweight class.

Proxy: FN: the Proxy class inherits the Service, a non-compliant pattern. FPs: see Decorator explanation.

Interpreter: FN: Interpreter method does not use the Context object for accumulating results. FPs: see Decorator explanation.

Iterator: FN: an external class was used as an abstract iterator; thus the overwriting of the abstract method could not be detected.

Mediator: FN: using functional programming; separating Listener and Handler classes rather than a common class for both purposes.

State: FPs: see Decorator explanation.

Strategy: FN: the client does not create specific strategies and does not define which strategy to use; decision made in the Context class, which does not reflect the classic pattern definition.

Visitor: FN: The Visitor methods, which apply different logic depending on the type of argument, are missing; there is only one method with the type of the parent class. This violates the pattern definition.

To summarize, out of 258 testcases there were 286 TPs, 17 FNs, 46 FPs, and 5585 TNs, resulting in 0.99 accuracy, 0.86 precision, 0.94 recall, and an F_1 score of 0.90. We believe this rate to be relatively good for application on this real-world sampling. 91 cross detections triggered a manual analysis with half of them being FPs. Due to their similarities, certain patterns remain challenging to differentiate based only on GA. Note that, despite being labeled as such on the internet, a number of the FNs were non-compliant or atypical implementations, affecting the accuracy rate. While these could have been culled beforehand, we wanted to utilize a dataset with real-world labeling. Having a larger benchmark dataset prepared or approved by experts in the future would be helpful for tuning. We note that the four lowest F_1 scores are for Proxy, Interpreter, State, and Adapter.

F. HyDPD-GA Evaluation (JUnit)

To evaluate DPD for a larger project, for **RQ6** we analyzed the latest version of JUnit source code version 5.8. The 1170 Java source files contained 83595 NCLOC (non-commented LOC) out of 143440 total lines. Since we do not presume to be familiar with the architecture of JUnit, we used a manual case-independent partial keyword search that included all the signal words used to train HyDPD-ML as well as certain other terms the GoF book contains with that pattern, including "also known as" or component or method names.

The results for the three GoF patterns to compare HyDPD-GA and HyDPD-ML are shown in Table X. Since the term "factorymethod" was found 137 times in 21 files, the range of HyDPD-GA and HyDPD-ML results seem probable. As adapter-related terms also occur relatively frequently, the range of results for HyDPD-GA and HyDPD-ML also seem probable. For Observer, since HyDPD-ML had the worst F_1

score with a precision of only 70.7%, in our opinion the high number of 689 hits are unlikely related to an actual implementation of the pattern and we would tend to see the HyDPD-GA results as more likely. If proved empirically true by further large project testing, one could, for example, tailor the HyDPD weightings of (1) for the Observer pattern more heavily towards HyDPD-GA.

TABLE X. DPD COMPARISON FOR JUNIT (THREE GOF PATTERNS)

Pattern	HyDPD-GA Hits [p<1]	HyDPD-ML Hits	Lexical Search	
			Keyword*	***Hits(Files) [raw]***
Factory Method	33 [24@0.71]	150	*factory*	972(185)
			create	1099(202)
			implements	417(267)
			type	3455(367)
			factorymethod	137(21)
Adapter	102 [13@0.75]	15	*adapter*	88(23)
			adaptee	8(3)
			target	538(145)
			adapt	100(24)
			wrapper	307(27)
Observer	0	689	*observer*	0
			state	204(39)
			update	0 [6(3)]
			notify	17(7)
			publish	222(58)
			subscribe	0
			subject	0
			attach	0 [3(2)]
			detach	0
			register	706(109)
			unregister	14(4)
			deregister	0
			setstate	0
			getstate	0

* partial any case code search; ML signal words in *italics* **[raw] values revised where obvious

For small results where it was obvious, raw values sometimes were adjusted if the search result context made it clear the pattern was not involved, e.g., the use of the word outside of a pattern context in a comment or in error handling code for a different purpose. This was in no way a systematic analysis of each search result.

HyDPD-GA was used for checking all 23 GoF patterns and provided various pattern detections that lexical analysis also found indicators for. Table XI compares our results with those of other work we found that published GoF DPD for JUnit, which utilized much older versions of JUnit. Nevertheless, we can compare the reported results by GoF pattern to see the relative extent of detection for such a project. Additionally, we performed a lexical search of JUnit v5.8 to determine if the pattern name as a keyword indicates possible usage, and this was marked next to our HyDPD-GA numerical result. While related work used only older JUnit versions, in comparison it does not seem to be completely off track in the detections, except for the high number of Adapter detections. As an explanation, for the three GoF patterns, HyDPD-GA had its lowest DPD scores for the Adapter and for all GoF its precision was 0.67 with an F_1 score of 0.77. Thus, the Adapter pattern is possibly confused and not necessarily as high, yet the lexical results in Table X may make the higher number possible relative to what related work had found.

TABLE XI. FULL GOF DPD COMPARISON FOR JUNIT

	HyDPD-GA	**Dwivedi** [28]	**Mayvan** [17]	**Oruc** [19]		**Yu** [16]	**nrp** [34]	**Sempatrec** [43]	**SSA** [44]
Year	2022	2018	2017	2016		2015	2014	2014	2006
Version	v5.8	-	v3.8, v4.1	v3.8	v4.1	v3.8		v3.7	v3.7
Abstract Factory	0*	6	0			0	0	0	na
Builder	11*					0			
Factory Method	33*		1			2	0	0	0
Prototype	1					0			
Singleton	7*		0	0	4	0	0	0	0
Adapter	102*	11	4			9	6	1	6
Bridge	3*	9		2	4	0			
Composite	0*		1	1	2	0	1	1	1
Decorator	1*		1	1	1	2	1	1	1
Façade	**					0			
Flyweight	1					0			
Proxy	1*					0			
CoR	0*					0			
Command	3*					0			
Interpreter	3					0			
Iterator	8*					0			
Mediator	1					0			
Memento	0					0			
Observer	0		3			1	3	1	1
State	0*		3			0	3	4	3
Strategy	0*					0			
Template Method	8*	38	1	12	22	1	1	1	1
Visitor	0*		0			0	0	0	0

*Manual lexical search indicates possible usage (may just use/extend Java API) **Memory issue

The results indicate that HyDPD-GA can be utilized on a larger project and potentially find or detect patterns. As the HyDPD-GA accuracy rates for GoF as shown in Section E above were relatively good, we expect the results for JUnit to be comparable. However, we note the issues mentioned in that previous section, where similar patterns that are mostly differentiated by intention can result in a different labeling to a similar pattern (e.g., Decorator, Adapter, Proxy, Interpreter, and State being similar in structure), being thus more easily confused and having lower F_1 scores. Detection would require a more in-depth analysis to determine if there are issues.

G. HyDPD Performance Evaluation

DPD performance was measured as depicted in Table XII for small projects (50 to 400 LOC from the test data sets) as well as for JUnit 5.8 (to exemplify a large project). The values are depicted on a log scale in Figure 8. The differences in latency are due to the varying number of positive (required) elements (nodes and relations) that need to be matched in a Cypher query while ensuring that negative unwanted elements are not in the structure. The queries thus vary in complexity and in turn affect latency. For instance, Interpreter has many conditions as well as negative conditions, whereas Singleton requires one class as a participant and has no negative conditions. The effects become more noticeable when analyzing larger projects.

TABLE XII. HYDPD-GA LATENCY

Pattern	Small project average (seconds)	JUnit 5.8 (seconds)
Abstract Factory	0.04	0.09
Builder	0.02	9.77
Factory Method	0.02	0.10
Prototype	0.02	0.21
Singleton	0.02	0.05
Adapter	0.04	183.71
Bridge	0.04	0.26
Composite	0.02	0.04
Decorator	0.02	23.81
Façade	0.02	error
Flyweight	0.02	1.41
Proxy	0.03	0.14
CoR	0.08	1.90
Command	0.03	12.60
Interpreter	0.02	692.98
Iterator	0.02	0.23
Mediator	0.02	0.07
Memento	0.03	1.59
Observer	0.03	3.23
State	0.02	3.01
Strategy	0.05	58.34
Template Method	0.02	0.21
Visitor	0.02	0.31
Total	0.63	994.06
Average	0.03	43.22

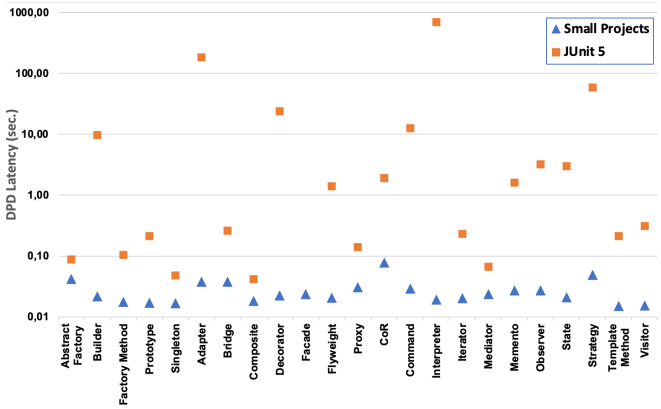


Figure 8. HyDPD-GA per-pattern latency: small project average vs. JUnit (log scale).

The total processing time needed for conversion, import, and DPD was measured as depicted in Table XIII. As one might expect for larger code bases, preparation processing time plays a more significant role, notably vector conversion for HyDPD-ML and Neo4j import for HyDPD-GA. During DPD execution, however, HyDPD-ML is not significantly impacted in contrast to HyDPD-GA.

To address performance for larger projects, one workaround might be to apply HyDPD-GA selectively for only certain pattern searches, or to apply HyDPD-ML initially since it executes much more quickly, and then selectively apply HyDPD-GA to certain patterns or only to certain modules to confirm those that HyDPD-ML detected.

TABLE XIII. TOTAL PROCESSING LATENCY

Process step	Small project average (sec.)		JUnit 5.8 (sec.)	
	ML	*GA*	*ML*	*GA*
scrML conversion	0.11	0.10	29.2	29.24
Training	5.90		5.90	
Vector conversion	16.75		22385.25	
MongoDB import		0.01		1.91
Neo4j import		0.08		948.48
Total preparation	22.74	0.19	22420.36	979.62
DPD execution	0.01	0.63	0.11	994.06
Total	22.75	0.82	22420.47	1973.69

H. Evaluation Discussion

The discussion of the evaluation results follows the **RQ**s:

RQ1: HyDPD-ML did demonstrate its feasibility, showing practical DPD results in cross-testing three common GoF patterns, with overall 90.7% accuracy, 87% precision, 86% recall, and an F1 score of 0.86.

RQ2: HyDPD-GA showed its feasibility, and performed relatively well against the three common GoF patterns, finding fewer but different overall fewer FNs and FPs than HyDPD-ML. Overall HyDPD-GA shows as good or better recall, precision, accuracy, and F_1 scores than HyDPD-ML, with the exception of the single FP for the Adapter pattern resulting in 96.4% vs. 100% precision, and the additional FN in the Observer pattern resulting in a recall of 93.3% vs. 96.7%.

RQ3: For the three common GoF patterns, the hybrid HyDPD demonstrated its feasibility, performing well with results of 94.4% recall, 98.8% precision, 97.8% accuracy, and an F1 score of 96.6%. In all cases, HyDPD provided as good or better results than either HyDPD-ML or HyDPD-GA alone.

RQ4: While HyDPD-GA can be useful for detecting more abstract architectural patterns, these are more challenging for GA to reliably detect due to their more abstract nature, enabling various implementation strategies. Testing the MVC pattern resulted in 0.80 recall, 0.62 precision, 0.53 accuracy, and an F1 score of 0.70.

RQ5: For checking HyDPD-GA against all 23 GoF patterns, cross-testing our 258 GoF testcases resulted in 5934 tests. It performed well, providing 0.99 accuracy, 0.86 precision, 0.94 recall, and an F_1 score of 0.90. It thus appears to provide quite useable results by itself. This could be especially suitable when larger datasets necessary for training (which HyDPD-ML would require) are unavailable.

RQ6: Based on the relatively large project JUnit (83K NCLOC), when comparing HyDPD-GA to HyDPD-ML for the three GoF patterns, a range difference in hits was observed, which correlates with our previous analysis that FNs of one component are often compensated as TPs by the other, or in other words, one DPD technique is better than another in certain circumstances. A lexical analysis of the code provided insights into the likelihood of the pattern usage, and the low precision of 70.7% for HyDPD-ML for the Observer pattern and the lack of clear lexical evidence would indicate it has a high FP rate for this pattern. HyDPD-GA performed relatively well. HyDPD-GA was used for checking all 23 GoF patterns and provided various pattern detections

that lexical analysis also found indicators for. While related work used only older JUnit versions, in comparison it does not seem to be completely off track in the detections, except for the high number of Adapter detections. As an explanation, for the three GoF patterns, HyDPD-GA had its lowest DPD scores for the Adapter and for all GoF its precision was 0.67 with an F_1 score of 0.77. Thus, the Adapter pattern is possibly confused and not necessarily as high, yet the lexical results in Table X may make the higher number possible relative to what related work had found.

RQ7: HyDPD-GA performance latency and scalability showed that for simpler queries its DPD performance in relative magnitude is on par with HyDPD-ML (for JUnit 5.8 a few seconds or less), but that particular patterns (in particular Builder, Adapter, Decorator, Command, Interpreter, Façade, Strategy) do require much longer query times. Preparation processing time plays a significant role before DPD can be executed, especially vector conversion for HyDPD-ML and Neo4j import for HyDPD-GA.

To summarize the evaluation, HyDPD has shown that it is viable for DPD for multiple programming languages. While combining the different strengths of HyDPD-ML and HyDPD-GA, HyDPD can also compensate for certain weaknesses of the other and improves the overall DPD capability (e.g., fewer FNs and improved F_1 score) while allowing for tailoring in weighting. More abstract architectural patterns such as MVC, while more challenging due to their abstract nature, can also be detected. For situations where insufficient training data is available for HyDPD-ML, HyDPD-GA can also be used alone and showed relatively good DPD results. Additionally, if performance and scalability are a primary factor, one alone can be chosen to lessen the impact on preparation or execution.

VI. Conclusion

This paper presented our HyDPD solution, a hybrid approach for generalized DPD utilizing graph analysis (GA) and Machine Learning (ML) and programming-language-agnostic approach to automate the detection of design patterns via source code analysis. Its realization demonstrates its feasibility, using srcML as a common markup language to support multiple programming languages, and its generalized approach works for many different patterns. The HyDPD-ML component was realized with TensorFlow using static code metrics and semantic analysis, while the HyDPD-GA component uses Cypher queries on the graph database Neo4j.

The evaluation compared each component and their combination for three common patterns across a set of 75 single pattern Java and C# public sample pattern projects. HyDPD-GA was also used to detect the 23 Gang of Four design patterns across 258 sample C# and Java projects as well as in a larger Java project JUnit. By applying the hybrid, HyDPD can compensate for certain weaknesses of the other component and improves the overall DPD capability (e.g., fewer FNs and improved F_1 score) while allowing for per-pattern or per-technique tailoring in probability weighting. More abstract architectural patterns such as MVC, while more challenging due to their abstract nature, were also detected. While HyDPD-ML requires sufficient initial training data for a pattern, HyDPD-GA can also be used alone without training and showed relatively good DPD results. Performance and scalability measurements showed the differences between components, which can be considered as to which technique to apply, with HyDPD-GA showing high performance-sensitivity for certain patterns due to the large number of matching and negative conditions that must be met.

Future work will investigate the inclusion of additional pattern properties and key differentiators to improve the results even further. This includes analyzing the network classification errors in more detail to further optimize the network accuracy, adding support for the remaining GoF patterns, utilizing semantic analysis with NLP capabilities on the code for additional natural languages, supporting additional programming languages such as C++. Also, we intend to evaluate pattern detection when they are intertwined with other patterns and address accuracy, performance, and scalability on large code bases. We will also investigate the detection of additional design and architectural patterns and implementation variants and integration with maintainer and developer tooling. Furthermore, to address the risk of overfitting, we intend to apply cross-validation and consider alternative classification schemes. Thereafter, we intend to do a comprehensive empirical industrial case study.

Acknowledgment

The author thanks Anna Kazakova, Florian Michel, and Christian Leistner for their assistance with the design, implementation, evaluation, and diagrams.

References

[1] R. Oberhauser, "A Machine Learning Approach Towards Automatic Software Design Pattern Recognition Across Multiple Programming Languages," Proc. of the Fifteenth International Conference on Software Engineering Advances (ICSEA 2020), IARIA XPS Press, 2020, pp. 27-32.

[2] M. Muro, S. Liu, J. Whiton, S. Kulkarni, "Digitalization and the American Workforce," Brookings Institution Metropolitan Policy Program, 2017. [Online] Available from https://www.brookings.edu/wp-content/uploads/2017/11/mpp_2017nov15_digitalization_full_report.pdf 2022.02.10

[3] C. Metz, "Google Is 2 Billion Lines of Code—And It's All in One Place." [Online] Available from http://www.wired.com/2015/09/google-2-billion-lines-codeand-one-place/ 2022.02.10

[4] [Online] Available from https://en.wikipedia.org/wiki/GitHub 2022.02.10

[5] G. Booch, "The complexity of programming models," Keynote talk at AOSD 2005, Chicago, IL, Mar. 14-18, 2005.

[6] [Online] Available from https://cybersecurityventures.com/application-security-report-2017/ 2022.02.10

[7] [Online] Available from https://web.archive.org/web/20210314184254/https://www.payscale.com/data-packages/employee-loyalty/least-loyal-employees 2022.02.10

[8] C. Jones, "The economics of software maintenance in the twenty first century". 2006. [Online] Available from https://web.archive.org/web/20160308070720/http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf 2022.02.10

[9] R. Minelli, A.Mocci, and M. Lanza, "I know what you did last summer: an investigation of how developers spend their time." In: Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension (pp. 25-35). IEEE Press, 2015.

[10] M.J. Pacione, M. Roper, and M. Wood, "A novel software visualisation model to support software comprehension." In: Proc.. 11th Working Conference on Reverse Engineering. (pp. 70-79). IEEE, 2004.

[11] A. von Mayrhauser and A.M. Vans, "Program comprehension during software maintenance and evolution," Computer, 28(8), pp. 44-55, 1995.

[12] E. Gamma, Design patterns: elements of reusable object-oriented software. Pearson Education India, 1995.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-oriented software architecture: a system of patterns, Vol. 1. John Wiley & Sons, 2008.

[14] L. Prechelt, B. Unger-Lamprecht, M. Philippsen and W. F. Tichy, "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance," in IEEE Transactions on Software Engineering, vol. 28, no. 6, pp. 595-606, June 2002, doi: 10.1109/TSE.2002.1010061.

[15] M.G. Al-Obeidallah, M. Petridis, and S. Kapetanakis, "A survey on design pattern detection approaches," International Journal of Software Engineering (IJSE), 7(3), pp.41-59, 2016.

[16] D. Yu, Y. Zhang, and Z. Chen, "A comprehensive approach to the recovery of design pattern instances based on sub-patterns and method signatures," Journal of Systems and Software, vol. 103, pp. 1-16, 2015.

[17] B. Mayvan and A. Rasoolzadegan, "Design pattern detection based on the graph theory," Knowledge-Based Systems, vol. 120, pp. 211-225, 2017.

[18] M.L. Bernardi, M. Cimitile, and G. Di Lucca, "Design pattern detection using a DSL-driven graph matching approach," Journal of Software: Evolution and Process, 26(12), pp.1233-1266, 2014.

[19] M. Oruc, F. Akal, and H. Sever, "Detecting design patterns in object-oriented design models by using a graph mining approach," 4th International Conference in Software Engineering Research and Innovation (CONISOFT 2016), pp. 115-121, IEEE, 2016.

[20] A. Pande, M. Gupta, and A.K.Tripathi, "A new approach for detecting design patterns by graph decomposition and graph isomorphism," International Conference on Contemporary Computing, pp. 108-119, Springer, Berlin, Heidelberg, 2010.

[21] P. Pradhan, A.K. Dwivedi, and S.K. Rath, "Detection of design pattern using graph isomorphism and normalized cross correlation," Eighth International Conference on Contemporary Computing (IC3 2015), pp. 208-213, IEEE, 2015.

[22] S. Alhusain, S. Coupland, R. John, and M. Kavanagh, "Design pattern recognition by using adaptive neuro fuzzy inference system," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 581-587, IEEE, 2013.

[23] M. Zanoni, F. A. Fontana, and F. Stella, "On applying machine learning techniques for design pattern detection," J. of Systems & Software, vol. 103, no. C, pp. 102-117, 2015.

[24] L. Galli, P. Lanzi, and D. Loiacono, "Applying data mining to extract design patterns from Unreal Tournament levels," Computational Intelligence and Games. pp. 1-8, IEEE, 2014.

[25] R. Ferenc, A. Beszedes, L. Fulop, and J. Lele, "Design pattern mining enhanced by machine learning," 21st IEEE Int'l Conf. on Softw. Maintenance (ICSM'05), IEEE, pp. 295-304, 2005.

[26] S. Uchiyama, H. Washizaki, Y. Fukazawa, and A. Kubo, "Design pattern detection using software metrics and machine learning," First International Workshop on Model-Driven Software Migration (MDSM 2011), pp. 38-47, 2011.

[27] S. Uchiyama, A. Kubo, H. Washizaki, and Y. Fukazawa, "Detecting design patterns in object-oriented program source code by using metrics and machine learning," Journal of Software Engineering and Applications, 7(12), pp. 983-998, 2014.

[28] A.K., Dwivedi, A. Tirkey, and S.K. Rath, "Software design pattern mining using classification-based techniques," Frontiers of Computer Science, 12(5), pp. 908-922, 2018.

[29] H. Thaller, L. Linsbauer, and A. Egyed, "Feature maps: A comprehensible software representation for design pattern detection," IEEE 26th international conference on software analysis, evolution and reengineering (SANER 2019), pp. 207-217, IEEE, 2019.

[30] A. Chihada, S. Jalili, S.M.H. Hasheminejad, and M.H. Zangooei, "Source code and design conformance, design pattern detection from source code by classification approach," Applied Soft Computing, 26, pp. 357-367, 2015.

[31] Y. Wang, H. Guo, H. Liu, and A. Abraham, "A fuzzy matching approach for design pattern mining," J. Intelligent & Fuzzy Systems, vol. 23, nos. 2-3, pp. 53-60, 2012.

[32] S. Alhusain, S. Coupland, R. John, and M. Kavanagh, "Towards machine learning based design pattern recognition," In: 2013 13th UK Workshop on Computational Intelligence (UKCI 2013), pp. 244-251, IEEE, 2013.

[33] S. Hussain, J. Keung, and A.A. Khan, "Software design patterns classification and selection using text categorization approach," Applied soft computing, 58, pp.225-244, 2017.

[34] A. Alnusair, T. Zhao, and G. Yan, "Rule-based detection of design patterns in program code," Int'l J. on Software Tools for Technology Transfer, vol. 16, no. 3, pp. 315-334, 2014.

[35] M. Lebon and V. Tzerpos, "Fine-grained design pattern detection," IEEE 36th Annual Computer Software and Applications Conference, IEEE, pp. 267-272, 2012.

[36] I. Issaoui, N. Bouassida, and H. Ben-Abdallah, "Using metric-based filtering to improve design pattern detection approaches," Innovations in Systems and Software Engineering, vol. 11, no. 1, pp. 39-53, 2015.

[37] Y. G. Guéhéneuc, J. Y. Guyomarc'h, and H. Sahraoui, "Improving design-pattern identification: a new approach and an exploratory study," Software Quality Journal, vol. 18, no. 1, pp. 145-174, 2010.

[38] F. A. Fontana, S. Maggioni, and C. Raibulet, "Understanding the relevance of micro-structures for design patterns detection," Journal of Systems and Software, vol. 84, no. 12, pp. 2334-2347, 2011.

[39] I. Issaoui, N. Bouassida, and H. Ben-Abdallah, "Using metric-based filtering to improve design pattern detection approaches. Innovations in Systems and Software Engineering," vol. 11, no. 1, pp. 39-53, 2015.

[40] J. Dong, Y. Zhao, and Y. Sun, "A matrix-based approach to recovering design patterns," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 39, no. 6, pp. 1271-1282, 2009.

[41] M. Collard, M. Decker, and J. Maletic, "Lightweight transformation and fact extraction with the srcML toolkit," IEEE 11th international working conference on source code analysis and manipulation, IEEE, 2011, pp. 173-184.

[42] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[43] G. Rasool and P. Mäder, "Flexible design pattern detection based on feature types," In 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 243-252, IEEE, 2011.

[44] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S.T: Halkidis, "Design pattern detection using similarity scoring," IEEE transactions on software engineering, *32*(11), pp. 896-909, 2006

Leveraging Gamma Corrections for an Overhead Reduced Mood Adaptive Display Coloring

Lukas Brodschelm, Felix Gräber, Daniel Hieber and Marc Hermann

Dept. of Computer Science
Aalen University
Aalen, Germany
Email: firstname.lastname@hs-aalen.de

***Abstract*—Humans can recognize a wide range of colors and interpret them in many different ways. Besides obvious effects like highlight and beautification, these colors can influence the emotional state of humans in a significant way. While this is no new information and color psychology is a heavily discussed topic in the psychological area, little research has been conducted in the human-computer interaction area of this topic. We presented a mood adaptive display coloring prototype in a previous paper, in this work the second iteration of the prototype is introduced. The second version extends the previous implementation, which utilizes psychological studies and state of the art machine learning technologies, with a new gamma shift-based coloring approach we present a greatly overhead reduced coloring service. The focus of using gamma shifting instead of overlays is to optimize the software for use on low-performance computers. To obtain equivalent results an approach is introduced, how to calculate a gamma-based shift, that is similar to alpha blend overlays. In order, that the overhead reduced version is still able to have the same influence effects on the emotion of human users as the previously presented prototype.**

Keywords*—*Advanced Human Computer Interaction; Mood Adapting Coloring; Adaptive Display Coloring; Color Psychology; Emotion Recognition.

I. Introduction

Colors have been an important part of Human-Computer-Interaction (HCI) since the introduction of multi color displays. Nowadays, it is impossible to imagine software without them, the old default black and white is only used as a stylistic instrument. Even hardware is often produced with ambient lighting as a selling point. Unconsciously perceived colors, like ambient lights or screen taints, can have impact on the users. In our previous work, we introduced a novel approach using these effects to influence the users' mood [1].

Colors are used as an important part of designs for Graphical User Interfaces (GUIs) in software like games, operating systems (OS) and business applications, either to help users to get the required overview or leading workers through complex procedures with intelligent coloring.

However, the possibilities of colors do not end with highlighting certain things and improving the quality of life; therefore, they can also be used to directly affect the user. Understanding the influence of color on the human user, clever coloring can also be used to change the users attitude, e.g., to get the users trust on a website [2].

That colors can have a quantifiable effect on the human behavior on special aspects is claimed in several publications [2]–[5]. The effect of colors on the mood of a person have also been studied but studies have not yielded clear results [6]. This is why color psychology is not only a highly disputed topic but also an an arguable research area. A possible reason for this might be, that the association and effect of color highly depends on the subject's background not only age and gender but also cultural aspects and probably many more have to be considered [3], [7]–[9].

However, thanks to the popularity of the topic much research is conducted in this area, allowing a solid foundation for further research in the HCI environment. Depending on this knowledge we introduced an approach to interact with users through decent adaptive coloring, providing emotional support and the best conditions to successfully master their current task in a preceding paper [1]. The software we developed for this is called MAD Coloring (Mood Adaptive Display Coloring), it is a display overlay framework, which taints the display color using an overlay, that is applied to the screen using an alpha blend algorithm.

The provided MAD prototype exposes a generic API for clients to control the color according to the mood of the user. Further two simple clients were implemented. One of them provides a minimal boilerplate for further implementations, the second one utilizes state of the art machine learning techniques to recognize mood changes and provide the best suited coloring solution for the detected mood state.

Using state of the art technologies, like facial recognition systems or intelligent devices such as smartwatches, the mood of a user can already be recognized and measured quite precisely. The resulting information can be used to classify the emotions and start processes to reinforcing or combating these.

In this paper, we introduced an alternative implementation to our last Coloring Service, replacing the overhead heavy Hudkit service with an overhead reduced gamma shift implementation. This includes, but is not limited to an updated configuration file handling, the connection to the X.Org display server and a transformation from RGBA based alpha blend to gamma correction, that ensures compatibility with already existing configuration files. Furthermore, we show the differences between the first, alpha blend overlay and the gamma based version.

The remainder of this paper is structured as follows: Section

II provides background information for a better understanding of the work and introducing required secondary research. Section III highlights the general concept of MAD-Coloring, while Section IV explains the currently implemented state of MAD-Coloring. Following Section V compares the newly introduced gamma shift Coloring Service with the old Hudkit service. In Section VI we introduce a possible Case Study to empirically evaluate our prototype. Section VII provides an overview of related works. Finally, Section VIII concludes the paper and lists further research options.

II. Background

This section outlines the elementary information about color psychology and display coloring frameworks, as well as face recognition basics and a survey about the emotional effect of color, conducted during this work.

A. Color Theory and Psychology

While color is often accepted as an omnipresent thing without further questioning, it is a well-defined construct which can be described accurately with its three components: hue, value, and chroma [10]. The emotional effect of color is closely linked to all three of those; therefore, an adjustment of one of its values can lead to a completely different emotional reaction. This means, e.g., a green with high brightness and saturation has a different emotional effect than a green with the same hue and saturation but low brightness.

Even though color psychology is highly disputed down to its fundamental theories, it is possible to elaborate some general statements that are used in this work. While many animals only see different shades of grey and some can even see with infrared or ultraviolet vision, humans have settled in the middle with complex color perception. This has lead to the development of a society where colors play an important role in the day to day life, religion, work, and free time. Most people have a favorite color, and every kid, regardless of their origin, knows the red cross, star or moon provides them with help if needed.

While most humans see color the same way, it has quite different meanings for them. The context in which a color is experienced and from whom is fundamental for its interpretation. Age is an important factor, as studies prove, that children have different associations with certain colors than adults or elder people. Gender can also play a decisive role [8][3][9].

Elliot and Maier laid the theoretical base for contextual color psychology with their work in 2012 [8], highlighting six key properties of the psychological impacts of color:

- There might be psychological relevant associations with a color.
- Presentation colors might influence psychological operations, including but not limited to basic impulses such as attraction and avoidance.
- Associations with colors might trigger affective, cognitive or behavioral reactions. This happens subconsciously or without intention.
- Color meanings and associations are influenced by both trained and inherited behavior.
- The relation between color perception and association is bidirectional. color perception has an impact on psychological processes and psychological processes have an impact on the way color is perceived.
- The psychological effect of color heavily depends on the context. The context is so important, that the influence of the same color might result in opposite effects for different contexts.

Those core statements are used by other research (e.g., [3]) in the field of color psychology and should be considered when targeting a psychological effect utilizing colors.

Li [9] gives a detailed overview of the preferences and effects with different groups of people in a medical context, more precisely during hospital stays. Children prefer brightly saturated colors and overall very colorful environments. These provide distraction from the tense situation and a calming effect, connected to the coloring, could be proven. Adults on the other hand tend to favor clean, cool colors like white, blue, and grey. They associate these with a professional environment and thus a higher chance of successful treatment. There is also a connected calming effect. However, the effect is achieved through the impression of a professional environment, that evokes the feeling of a qualified treatment when seeking medical care. Therefore, it is only partially connected to the color and already an effect of the original feeling emitted by the color: professionalism. The author further states, that elder people tend to prefer warmer colors, although not as bright and less saturated than children. These colors help them to relax and effectively reduce anxiety. The professional, cool colors like white or grey even provide a negative effect on elders and sometimes children. This originates from the partly occurring associated with anxiety, loneliness, and fear. The phenomenon is known in color psychology as ”white scare”.

Focusing on the target-context oriented effect of color, Maier et al. outline how the psychological impact of color changes with the task domain [5]. Stated are the findings of different studies which point out that colors provide a performance-enhancing effect in physical competitions. An example of this can be found in sports, where teams wearing red tricots win more games than those wearing any other hue. However, when it comes to an intellectual target-context the color red seems to have no, or even a negative effect, on the performance.

B. Emotion Recognition

Facial and Image Recognition is currently one of the most popular fields of machine learning. Elementary face detection is not a technical challenge anymore and a camera input can be analyzed in realtime, with only a few dozen lines of code (e.g., [11]).

The popularity of such use cases provides us with the base for emotion recognition through feature detection in the mimic of people’s faces. Combining this technology with display coloring frameworks allows us to design systems that are

capable of autonomously detecting emotions and correctly using different color overlays to improve and reinforce them.

Several models have been trained with fairly high accuracy to detect emotion from images. These models focus on a combination of facial features to detect the emotional state behind pictures of faces. The model of Yang et al. [12] relies on the extraction of the mouth and eyes, achieving successful detection rates of up to 93% for certain emotions and a mean of 87%.

As emotions have a broad impact on the users' behavior the detection is not only limited to image recognition. For precise detection of emotions, multiple sources of information can be used. Ghosh et al. suggest an approach to determine emotion based on speech recognition [13]. By combining multiple ways of emotion recognition detection accuracy can be improved even further. However, background software with recording capabilities is rather problematic from a privacy point of view. Therefore, the improved recognition accuracy does not outweigh the privacy violation.

Other possibilities to increase the accuracy and therefore the value of automatic emotion detection could be provided by smart wearables like smartwatches or fitness trackers. Measurements like the pulse could then be used to determine the stress level and provide other valuable insights. However, as the prototype should be as lean as possible this approach is not further considered in this work. The required multi machine solution with means of communication between devices would cause a too excessive codebase for the aspired simple prototype.

C. Emotional Effects of Colors - a Survey

As our excessive study of related research projects and studies could not provide a common interpretation of the concrete connection between emotions and colors, a simple survey was conducted. This survey, however, is only used to generate a default profile for the prototype, it is not scientifically representative. In the survey, 522 people were asked to think of color if confronted with an emotion. Multiple answers were allowed, this should allow the detection of patterns, e.g., all warm colors are connected with emotion X. The emotions which could be chosen in the survey were (Ca)lm, (V)italising, (Sa)fety, (Co)ncentrated, (M)elancholy, (St)ressed and (H)appieness. The possible answers consisted of colors which can easily be displayed on a display and provide strong contrasts to each other, allowing a meaningful implementation.

The survey's findings match well with the general statements from Rider [3]. While warmer colors (orange, yellow) have an arousing effect, cooler colors (blue, green) have a relaxing, calming effect. Rider further mentions the widespread of possible emotions connected to green depending on brightness, saturation, and context. As we only asked for the hue the subjects could not differentiate between different types of green. Therefore, green is strongly connected with the most positive emotions, which, however, most likely refer to different types of green.

TABLE I. Emotional Connection to Colors Survey, Respondents = 522, values in %

Color	Ca	V	Sa	Co	M	St	H
red	3,79	14,86	13,38	7,32	9,37	**35,20**	8,45
orange	8,24	**19,41**	**18,09**	6,02	3,02	16,94	13,49
yellow	3,53	**21,15**	7,21	8,95	3,78	14,31	17,53
green	**22,09**	**18,34**	**15,44**	15,49	2,27	2,96	**24,72**
blue	**25,23**	9,64	**16,32**	**28,74**	7,55	4,77	14,25
violet	11,37	3,35	11,32	5,34	11,48	5,76	6,56
grey	11,11	0,40	5,44	13,08	**27,49**	3,95	2,27
black	12,81	2,68	10,29	13,43	**29,61**	6,74	4,41
pink	1,83	10,17	2,50	1,72	5,44	9,38	8,32

The survey results also concur with the findings of Kido [4], reinforcing the calming effect of green and blue. The deafening effect of red, determined by Kido, also matches the very strong association with stress by our survey.

The strong association of orange with safety, which cannot be found in other studies or surveys, most likely is caused by a bias of our survey. The poll was conducted on a group of people from the same community, where orange is a frequent and positively interpreted color, which could explain this behavior.

D. Display Overlays

Display overlays are no new research topic and many refined products are available. In the gaming area, they are omnipresent with the most popular example being steam's in-game-overlay, other popular examples being the Nvidia Geforce Experience overlay, the Discord overlay or Overwolf. However, these are all limited to the gaming use cases and only Overwolf allows great modification freedom.

Overlays might either display solid colors or transparent graphics on top of other applications. When using unicolored transparent images, defined by a red, green, blue, and alpha channel (RGBA), the screen is tinted with the defined color. This procedure, of combining two images, is known as alpha blending or alpha compositing [14]. However, there is not just one but many ways to combine two images. Therefore, there are several formulas, which might be used to calculate the resulting color for the alpha blend.

The pqiv image viewer can display transparent pictures and can easily be placed on top of other windows [15]. Due to its implementation in Python, cross platform capabilities are provided. However, the lack of click-through support requires extensive extension work to create a functional, non-blocking overlay.

OnTopReplica, a C# project, supports the display of selected windows on top of others on Windows systems [16]. While it supports click-through and adjustable opacity, it is limited to Windows and requires an additional overlay-window. This not only decreases the compatible systems but also increases the overhead significantly.

Hudkit is a C based framework with an exposed JavaScript-API [17]. The framework supports all Linux X-desktops and some OS X systems. Its main HTML page can be modified like

Overwolf overlays using HTML and JavaScript. The page can then receive new input via established APIs like Websocket or WebGL and change the display accordingly. Multiple monitors and click-through events are supported by default. Providing a powerful small footprint framework.

E. Gamma Correction

Gamma correction is a luminance adjustment technique, designed to provide color genuine images on common computer monitors. The gamma(γ), which is added to the luminance of the displayed image relies on an exponential scale, the resulting color value is usually calculated for every pixel using the formula from Equation (1) (cf. [18] Advanced Lighting → Gamma Correction).

$$\vec{C}_{RGB} = \begin{pmatrix} Pixel_{red}{}^{\frac{1}{\gamma_R}} \\ Pixel_{green}{}^{\frac{1}{\gamma_G}} \\ Pixel_{blue}{}^{\frac{1}{\gamma_B}} \end{pmatrix} \quad (1)$$

Where gamma might be any floating point value between 0 and 10. This exponential approach is chosen as the human eye senses changes in the lower brightness spectrum more than those for bright colors (cf. Foley et al. [19] pp. 448-449).

Gamma corrections are freely available to the user on the major operating systems. On Microsoft Windows via DXGI [20] and on Linux Systems via the X.Org display service utilizing APIs like Xlib [21]. Apple's macOS also provides some methods to manipulate gamma, however, they lack sufficient public documentation. The gamma correction is configured within the display settings and is applied to the visible screen by the display service or operating system without the need to run any additional third party software.

On X.Org based Linux desktops this gamma correction might be configured via the Xlib API or one of its corresponding bindings, providing programmatic access to the X11 display service. However, the library interface itself is quite extensive and tools like Xrandr, which provides the functionality of the Xlib and its libXrandr extension as a command line tool provide a much more simplified interface [22].

Xrandr allows the user to adjust the gamma correction for every color channel via the command line option: -gamma R:G:B. By applying values larger than 1 the luminance is increased, by applying smaller values the luminance is decreased. When setting different luminance values for the different channels a color shift is applied to the screen.

III. Concept

This section provides insights into the architecture and concept for the current MAD-Coloring prototype. The modular architecture approach is shown in Fig. 1.

The architecture is separated in 3 sections. The Coloring Service, the clients accessing and controlling the coloring and the interface, enabling communications between the clients and the display coloring.

The display coloring has to be able to project a color adjustable overlay on top of all connected monitors. While we

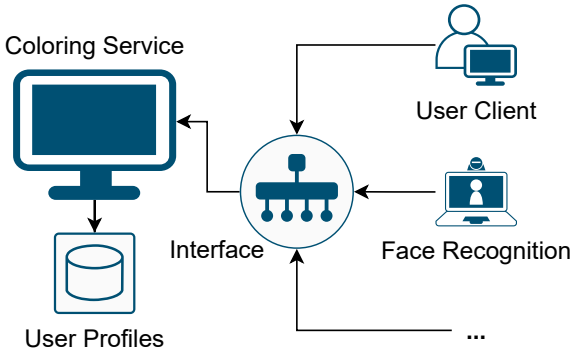


Figure 1. MAD-Coloring modular architecture concept.

stated in our previous work [1] that gamma corrections are not sufficient we have to stand corrected. While linear gamma or brightness adjustments are not enough to fully leverage the psychological effects of the overlay, a calculated vector based gamma shift as introduced in this paper is capable of producing promising results. The display must not reduce the productivity and therefore has to enable all actions which are possible without an overlay, e.g., right/left clicking or text selection. Further, the vision quality and readability of displayed content must not be reduced any further than necessary. This will be compared against the light constraints coming with night lights, e.g., reduced contrast and slightly changed colors. The Coloring Service further gets supplied with at least one user profile file. This allows users to adjust the color optimally for themselves and maximize the effect of MAD-Coloring.

The communication between clients and coloring should be kept as simple as possible to fit into the modular approach. This should enable easy access to the Coloring Service for new clients as well as the replacement of the Coloring implementation itself, e.g., if required by a change of the OS. To achieve this a simple interface should be created, supplying all required functionality for the coloring and client programs. New implementations could then simply use this interface and would be able to be used with the old clients/Coloring. This further enables the exchange of the communication framework with changes to only the interface itself. Clients and coloring would not be affected and could be used without any changes. The underlying communication framework should be chosen OS specific and use existing infrastructure rather than implementing something new. Therefore, reducing the implementation effort, requirements and overhead caused by MAD-Coloring, compared to an out-of-the-box solution shipped with a communication framework.

Due to the interface, client programs and Coloring Services can be created with minimal restrictions. For the prototype two client applications are planned. A manual user client, allowing the selection of moods by the user via text input, and a face recognition background service, detecting the mood via a webcam and changing the display coloring accordingly. Further, for the current prototype two Coloring Services are planned. Additionally to the already existing Hudkit-Coloring-

Service from our previous work a Gamma-Shift-Coloring-Service is planned, utilizing a vector based gamma shift. Following our modular architecture approach, the configuration files/user profile from one service can be used in the other service.

Thanks to the highly adjustable user profiles, changes or fine tuning of the color mapping in the implementation are always possible with little to no programming experience. This allows interested psychologists, therapists, and doctors to use the MAD-Coloring on their own. Enabling them to adjust the system according to their knowledge and research in the area of color psychology, highly tailored to their target group needs. Further, users trying the prototype on their own can adjust their profile as they most see fit, according to their preferences.

IV. MAD-Coloring

This section gives insight in the implemented MAD-Coloring prototype for Linux systems, including the different clients and Coloring Services. While some parts of it are Linux specific, like the specific D-Bus interface or gamma handling via Xrandr, the solution can easily be ported to Windows or macOS systems with small changes to the specific modules.

A. Interface

The interface is built on top of the D-Bus, as it represents the default solution for inter-process communication of most Linux desktop environments and therefore already is available on the system. This removes the need to install new software frameworks, perfectly fitting into the low overhead architecture of the concept.

The interface itself is separated into two parts. The main (top level) interface, providing simple python functions and a bottom level interface consisting of a D-Bus service and a corresponding client.

To ease the use of the interface and allow easy interchangeability of the underlying system the top level interface has been implemented only exposing the core functionality to client developers. It can be considered as a wrapper around the bottom level interface, providing the two required functionalities to implement new clients or a Coloring Service. These are a getter and setter functions for communication with the clients.

While the top level interface is rather simple, the bottom-line D-Bus communication is more extensive. The D-Bus service specified by it exposes an interface with three methods on the D-Bus session bus. However, only one of the three methods is currently used in the top level interface, a function to set the mood. The other methods provide interfaces for the currently active color code and the possibility to change the user profile on the fly, providing a boilerplate for more extensive clients. For the top level interface setter function, a connection to the exposed D-Bus interface is opened and the provided bottom level interface setter method is used to send the color change via the D-Bus.

It is up to the client if the communication is done via interface provided in this work or directly via the D-Bus. On the one hand, the baseline D-Bus service provides a richer API and might be used with any programming language or due to some universal command line interface tools. However, on the other hand, using the Python interface instead is less complex and allows an easy exchange of the underlying D-Bus framework, which might be required due to the change of the OS. Requiring only the getter and setter pair at the top level interface allows an easy exchange of the base interface, as the new framework must only realize these two functions. When sticking to Linux Desktops the D-Bus provides more extensive functionality and grants the exchange of the programming language, but the provided python interface grants interchangeability of OS and base interface. Choosing the right interface is up to the developer and should be decided individually according to the needs of the implementation.

B. User Profiles

The profiles are implemented as .conf configuration files. Using this well known and easy to use standard allows the modification of these files without any knowledge of programming. This is important, as for now, the prototype provides no GUI to edit the user profiles and the users have to edit their profiles with a text editor on their own.

In the scope of the prototype, the emotions and their respective coloring in the default profile are defined according the survey in Section II, which is shown in Listing 1. If the users want to change the color of an emotion they can either change the default profile or create a new user profile. Following an override approach, the Coloring Service will always check the specific user profile first. If the emotion is not defined in this profile, the service will fall back to the default profile. This makes it quite comfortable to specify some custom mood/color pairs while keeping default settings for most.

Listing 1. Example color definition

```
[Colours]
tired = #ede35a33
stressed = #5aeded20
sad = #35e85620
angry = #49d2fc20
unconcentrated = #a4f6fc20
anxious = #e8a63533
frustrated = #5aeded33
bored = #e8413555
```

It is also possible to define new emotion/color pairs in the user profiles, which are not defined in the default profile. However, to be used by the clients, the emotions have to be added to their codebase as well.

With the Gamma-Shift-Coloring-Service the user profiles were extended to also handle gamma shift values rather than only RGB values. This is detailed further in the Gamma-Shift-Coloring-Service section.

C. Hudkit-Coloring-Service

In the first version of our Coloring Service a combination between a modified version of Hudkit, introduced in Section

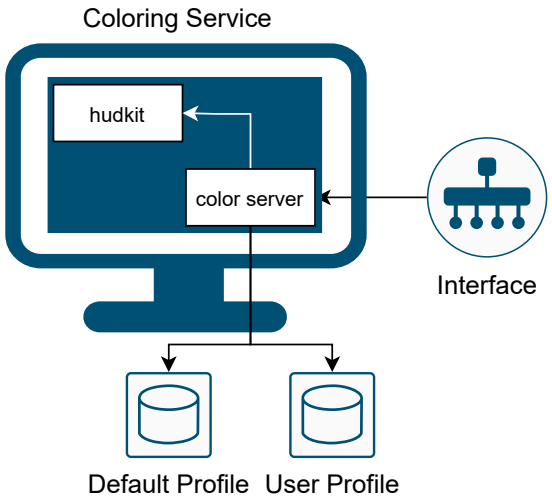


Figure 2. MAD-Coloring color service.

II, and a Python Color server is being used (Fig. 2).

Hudkit completely handles the display coloring. This happens by blending an RGBA colored image over the screen to taint it in the required shade. The alpha(α) blend of the display overlay is done via a linear calculation for every pixel on the screen. We define the tainting colors in the user specific color profile in hexadecimal encoded RGBA values. The color, which is displayed on the monitor is calculated using the "normal" color and the blend of the overlay, cf. Equation (2).

$$\vec{C}_{RGB} = \begin{pmatrix} Pixel_{red} \\ Pixel_{green} \\ Pixel_{blue} \end{pmatrix} * (1-\alpha) + \begin{pmatrix} Taint_{red} \\ Taint_{green} \\ Taint_{blue} \end{pmatrix} * \alpha \tag{2}$$

The color server provides the color services interface and control unit. This is achieved by implementing the earlier introduced Python interface and accessing the user profiles. Utilizing the Python interface, the server listens to changes of the mood send by a client. If a mood change is detected the server resolves the color according to the used profile. First, the user specific configuration is read, if no entry for the specific mood is found the default configuration is read.

After resolving the color, the server sends a signal with the determined color to the Hudkit web server. Should a mood not be defined in the configuration files, the color server defaults to black with low saturation and displays an error. With this error handling a missing emotion does not lead to further problems during the runtime, but clearly signals the user that a problem with the used client has occurred. This error color will not be interpreted as an emotion by the user, as black is not used by the default configuration and we highly discourage the usage of black as it is mostly related to negative emotions (cf. Table I). To counter possible negative emotions associated with black, the low saturation and higher brightness creates a grey overlay in the Hudkit server. This can be interpreted by users as boring, but will not trigger negative emotions.

D. Gamma-Shift-Coloring-Service

In this paper, we introduce a second kind of Coloring Service for our prototype where the display overlay Hudkit is exchanged against color shifting implementation utilizing the X.Org gamma correction via Xrandr, introduced in Section II. By doing so, we omit the display overlay software and reduce the performance overhead of the prototype.

The X.Org display service is handling the configured gamma correction. In our scenario, this gamma is configured via the Xrandr API. It is called from within the Python Coloring Services color server, after calculating the gamma correction for the configured color shift. We chose to use the binary Xrandr for the prototype rather than LibXOrg itself or its corresponding Python bindings, as the refined API provided by Xrandr greatly reduces the implementation complexity.

In order to follow the modular exchangeable architecture of MAD-Coloring and keep the same configuration files, an conversion from RGBA values to a gamma adjustment is required when using gamma correction instead of alpha blending. The major difficulty here is, that alpha blending is perceived linearly, while gamma correction is perceived exponentially by the human eye. Further, while alpha blending is displayed as an overlay, independent of the colors behind it, the gamma correction is highly dependent of its actual background. This means, that it is not possible to compute an overall gamma, which has the same effect for every background color as the corresponding alpha blend.

However, it is possible to calculate a gamma correction, which has the same effect as a RGBA alpha blending on a fixed background color via a RGB vector shift. The general formula for a vector based gamma shift on a single pixel can be seen in (3). The derivation for this equation can be found in Equation (8) (Appendix).

$$\vec{\gamma}_{RGB} = \begin{pmatrix} \frac{1}{log_{Pixel_R}(Pixel_R + (Taint_R - Pixel_R) * \alpha)} \\ \frac{1}{log_{Pixel_G}(Pixel_G + (Taint_G - Pixel_G) * \alpha)} \\ \frac{1}{log_{Pixel_B}(Pixel_B + (Taint_B - Pixel_B) * \alpha)} \end{pmatrix} \tag{3}$$

Calculating the correct gamma value for each pixel and coloring it accordingly would be absurdly resource intensive, anyhow it is possible to compute a general approximate γ value by using a fixed color vector. Determining the optimal base color vector for this use-case still needs further work, however, using the neutral grey (50% grey) color vector as displayed in Equation (4) provides an interim solution with adequate results.

$$\vec{C}_{50\%-grey} = \begin{pmatrix} 128 \\ 128 \\ 128 \end{pmatrix} \tag{4}$$

While using a fixed color vector as a base is no perfect solution, it is the most resource saving and yet functional approach. Resolving Equation (3) with Equation (4) we receive the formula used for the calculation of our vector based gamma

shift, introduced in Section III, which can be seen in Equation (5)

$$\vec{\gamma}_{RGB} = \begin{pmatrix} \frac{1}{log_{128}(128+(Taint_R-128)*\alpha)} \\ \frac{1}{log_{128}(128+(Taint_G-128)*\alpha)} \\ \frac{1}{log_{128}(128+(Taint_B-128)*\alpha)} \end{pmatrix} \quad (5)$$

When looking up a color, the Hudkit-Coloring-Service transmits the color via the color server directly from the user profile to the display overlay. The Gama-Shift-Coloring-Service newly implemented in this paper first has to calculate the corresponding gamma shift value using the formula defined above.

In X.Org based systems the gamma correction is by default set to 1.0. For the Gamma-Shift-Coloring-Service we assume, that no changes were made to this default gamma correction by the user via system settings. However, we extended the color server and user profile with an option to define a default gamma correction value (red/blue/green-pre) which will then be taken into account for the calculation of the vector based gamma shift. Under the premise that the gamma shift from the Coloring Service is applied before any gamma correction defined in the system settings, the resulting screen color may be computed via the formula from Equation (6) representing the final formula of the Gamma-Shift-Coloring-Service. This formula is derived by inserting the resulting vector based gamma shift from Equation (5) as the pixels color value in the color vector formula for gamma from Equation (1). For better readability, a vector free display form was chosen.

$$\begin{aligned} \gamma_{red} &= \frac{1}{log_{128}((128+(Taint_{red}-128)*\alpha)^{\frac{1}{\gamma_{red-pre}}})} \\ \gamma_{green} &= \frac{1}{log_{128}((128+(Taint_{green}-128)*\alpha)^{\frac{1}{\gamma_{green-pre}}})} \\ \gamma_{blue} &= \frac{1}{log_{128}((128+(Taint_{blue}-128)*\alpha)^{\frac{1}{\gamma_{blue-pre}}})} \end{aligned} \quad (6)$$

In the context of the Gamma-Shift-Coloring-Service it might be confusing to set RGBA values in the user profile configuration file since they are not applied directly as with the alpha blend method from the Hudkit-Coloring-Service. We, therefore, added the option to define the emotion-color pair in the user profile with a gamma shift instead of a RGBA value. To extend the new usability feature with full compatibility to our old user profiles, we enhance the color server with a detection mechanism. The algorithm recognizes if an entry in the user profile configuration file contains an RGBA value or a gamma correction shift and accordingly handles the value. The Hudkit-Coloring-Service, however, still requires RGBA values and does not work with gamma shift values. Further extensions to the user profiles and coloring servers are planned, to overcome the current limitations.

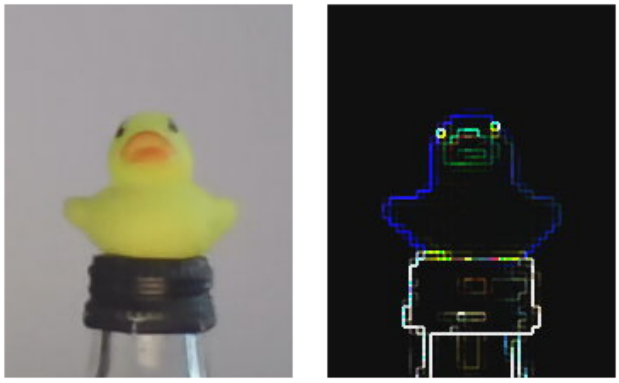


Figure 3. Face detection symbol image - feature highlighting.

E. Clients

The prototype includes two clients. A simple user client and an intelligent emotion detection client. The simple client provides a basic GUI to enter the current emotion and change the display coloring accordingly. It is implemented in Python and directly accesses the D-Bus instead of the Python wrapper interface.

The second client is an emotion recognition client, detecting the current emotional state of the user via image processing of a webcam feed. As a foundation the work of Rovai was used, creating a face recognition system utilizing a webcam with Python and OpenCV [11].

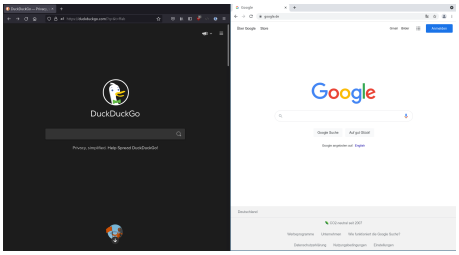
However, the final implementation differs fundamentally, as the client was extended to enable emotion recognition and connected to the Python interface to communicate with the Coloring Service. Further, a multi face detection was implemented, preventing a flickering color change if two or more faces are detected. As a result the client will not send emotional changes to the Coloring Service, until only one face is left in its field of view. A simplified version of the used feature detection can be found in Fig. 3. The eyes are clearly detected in white and the mouth in green. This allows the usage of the features in the neural network to determine the mood.

Due to the clean interfaces, between client and service, both clients are fully compatible with both Coloring Services versions. Therefore, no changes to the clients were necessary.

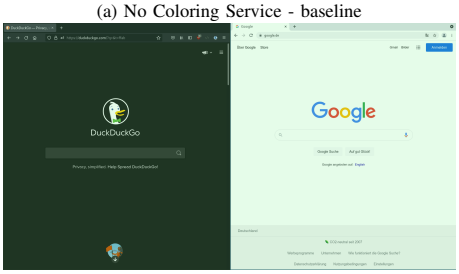
V. Comparison of first and second prototype

This section provides a quick overview between our current prototype and the version introduced in our previous work [1]. The focus is therefore centered around the new Gamma-Shift-Coloring-Service and the required introduced changes.

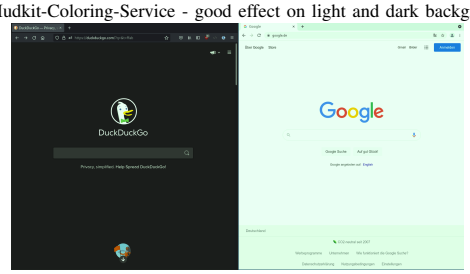
When comparing the first version of the prototype, which provided a display overlay utilizing alpha blending (Hudkit-Coloring-Service), with the second version, major graphical differences are visible. Since gamma correction is an exponential scaling algorithm the impact on the result depends strongly on the underlying colors. Figure 4 highlights the



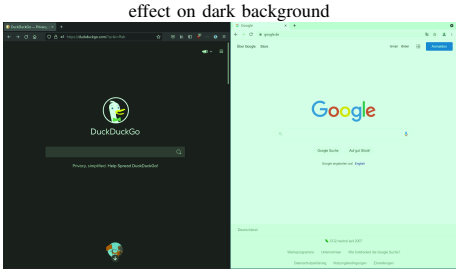
(a) No Coloring Service - baseline



(b) Hudkit-Coloring-Service - good effect on light and dark background



(c) Gamma-Shift-Coloring-Service - good effect on light background, minimal effect on dark background



(d) Gamma-Shift-Coloring-Service double opacity - extrem effect on light background, small effect on dark background

Figure 4. Comparison of different MAD-Coloring Color-Service effects for sad mood

comparison between applying alpha blending and using a gamma correction shift on dark and bright background images. In the images, it becomes clear, that the alpha blend takes much more effective to black areas than the vector based gamma shift approach does.

Considering the implementation aspects of both prototype versions, the technical differences are limited to the Coloring Service, which defines how the color adjustment is displayed and the configuration reader, as well as the connected user profiles. On the one hand, the configuration files and their parsing has become slightly more complex in the second version of the prototype, to enable the configuration and usage of gamma values. On the other hand, the inter-process communication procedure of the displaying interface as well as the overhead of the Coloring Service has been greatly reduced, making the whole MAD-Coloring application more lightweight. As a downside the calculations for the vector based gamma shift are significantly more complex, however, this only concerns the implementation of the prototype and has no negative effect on the end user.

Taking the technical and visual changes into account the vector based gamma shift provides a solid option for computers with lower tech-specs, allowing a more resource-conserving execution of MAD-Coloring. To counteract the weaker coloring effects of the gamma shift on dark backgrounds, compared to the alpha blend method, the usage should be limited to applications in light-mode, e.g., Excel, Office or browser usage. Summarizing, the vector based gamma shift Coloring Service provides a great solution for weak office machines, while the alpha blend Coloring service is the all-rounder with higher overhead, e.g., for gaming use-cases.

VI. Case Study

Our current team solely consists of researchers from the IT domain, we have not yet consulted any psychological/medical experts. Thus, we conducted a technical case study instead of a psychological evaluation of our prototype. The study was conducted on a Lenovo ThinkPad with 14GB RAM, an AMD onboard graphic chip, using a GNU/Linux operating system with a gnome X.Org desktop. During the test runs checks regarding usability impairments were conducted.

To pre-empt ethical concerns without the approval of an ethic-committee our test user simply controlled MAD-Coloring and rated possible concerns regarding the readability and usability of the desktop with activated MAD-Coloring. However, he was not exposed to the software for longer periods. The face recognition was triggered with prepared photos instead of live images of the test user.

As a scenario, a computer science student with no prior experience of MAD-Coloring was instructed to start the MAD coloring client and conduct a manual color change via the simple user client. Afterward the face recognition client had to be started. He was supplied with a computer that had a preinstalled MAD-Coloring Service and the MAD-service's Linux manual page.

The user was able to activate the service, change the color and start the face recognition client in less than a minute. All color changes and emotion detections worked without any problems. The user did not experience any problems regarding reduced readability, yet the usability in color sensitive applications like image editing was highly impaired because of MAD-Coloring's color overlay. However, this was an expected side effect as mentioned in our concept (cf. Section III). The effect of MAD-Coloring for some emotions can be seen in Fig. 5.

Following, we suggest a method for measuring the effects of MAD-Coloring in coming evaluations, which still needs to be reviewed by psychological experts. Since MAD-Coloring is designed to be used in daily routines and any kind of test scenario might cause discrepancies, we strongly recommend to evaluate MAD-Coloring integrated into the daily life of subjects.

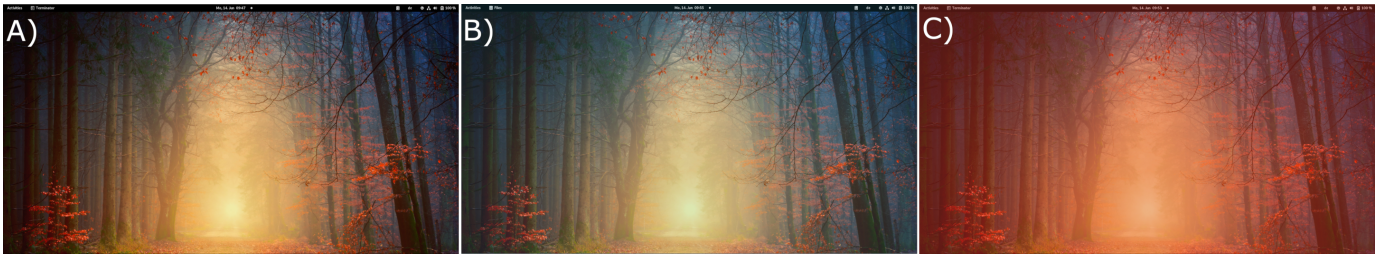


Figure 5. MAD-Coloring in effect for moods A) neutral B) mad C) bored.

To quantify MAD-Coloring, we introduce the concept of a mood diary where subjects record their emotional condition, and whether they are able to concentrate for work or not. The subjects might write this mood diary for a reasonable period (2-4 weeks). Afterwards, based on these diaries, basic emotional profiles can be created for all subjects.

In the second phase we recommend to split up the subjects into three groups. Group A will be working with our MAD-Coloring and the pre-configured color profile (based on scientific work from the color psychology domain and our own survey). Group B is working with MAD-Coloring as well, but with ”anti”-colors, which have been associated to be negative according to a specific mood (e.g., red if the subject is already mad). Group C is the control group, which continues working without any influence from MAD-Coloring. This phase should be conducted over a larger time period (2-4 months or longer), as most likely some time is required to get used to the color changes. Especially at the beginning these changes could have a negative impact on the subjects.

By comparing the deltas of the three groups the essential effects of MAD-Coloring can be determined and whether the effect depends on specific colors or just generally on shifting these. Afterwards further evaluations can be planned targeted on the existing data.

VII. Related Research

While the idea of color psychology is not new, there is to our best knowledge no closely related research in the HCI context. However, some other research topics in the context can be viewed as relevant.

A. Blue Light Filtering

At the first sight blue light filtering software seems to differ quite significantly from our solution, but the fundamental ideas are quite similar. Both software solutions modify the color shade of the display to obtain effects on the human user. However, blue light filtering is based on different medical effects (cf. [23]) and in most cases it is implemented completely different than MAD-Coloring.

While MAD-Coloring is based on the psychological effect of colors and the emotions triggered by colors, the idea of blue light filtering, as described in [24] is based on physical and bio-chemical effects [25]. They determine that blue light emitted by screens contains more energy than any other color. Further, it is more exhaustive for the human eye than other colors. In addition to the physical aspects blue light suppresses the production of the sleeping hormone melatonin which can cause sleeplessness.

As aforementioned, blue filtering software is often implemented completely different than our solution. Its common among blue filters to adjust the alpha channel to suppress parts of the blue light. The resource costs for this approach can be expected to be less than those for overlay based filters. This is due by the fact, that the graphics card is not required to compute a translucent overlay in. However, for our MAD-Coloring system an alpha shift approach does not fit the requirements since the software needs to display tints in various colors.

B. Colors and Trust in User Interface Design

Hawlitschek et al. [2] thematize the influence of colors on trustworthiness of user interfaces. They analyzed the moods and meanings associated to different colors via an experiment.

In this experiment, the probands had to pass a finance based trust experiment. They were provided with GUIs in different hues and small amount of money was handed to each proband. Then, by transferring money to other probands, they where able to increase the value of their sum by trading between each other. The experiment tried to determine if the color of their GUI had an impact on the trust they have in the other probands. However, they where not able to gain meaningful results from this experiment.

C. Effect of Colors on Emotions in Games

Joosten et al. [26] empirically studied the emotional effect of color on players of a video game. As a foundation they used a subset of the color⇒emotion arrangements of Plutchik [27], allocating the emotional effect of colors to an emotion as following; dark green ⇒ fear, light blue ⇒ surprise, red ⇒ anger, and yellow ⇒ joy.

The evaluation was conducted on 60 participants which all had to play five levels of a video game programmed for this evaluation. The first level acted as a control level, being the same for all players, while the following four levels had 24 possible deviating color arrangements. These were evenly distributed on the players, leaving 2-3 players per arrangement.

With this setup the expected negative arousing effect of red, as well as the positiv effect of yellow could be confirmed, while the other two colors did not had a meaningful effect on the players emotional state.

VIII. Conclusion

In this paper we presented the second version of our novel MAD-Coloring framework, highlighting a Hudkit-free Coloring Service. Respecting the basics of color psychology a second, more lightweight, Coloring Service prototype was implemented, providing a transparent, color adjustable overlay for Linux X-desktop systems utilizing a vector based gamma shift approach. This new service was tested with our two earlier clients. A simple input client, allowing the manual change of the display color and an emotion recognition client, detecting the users current emotional state via a webcam and adjusting the display color accordingly. We also extended our user profiles with options to directly enter gamma values, to increase the ease of use with the new Coloring Service.

MAD-Coloring, in combination with this clients, is capable to display a decent, transparent overlay over multiple desktops according to the users current emotion. The new gamma based version of the Coloring Service provides similar results as the initial alpha blend version. However, the coloring over dark backgrounds is harder to notice, potentially reducing the effect. Then again, the technical footprint of MAD-Coloring was greatly reduced with the new gamma shift approach, removing the overhead introduced by Hudkit.

While this version of the Coloring Service provided overall slightly worse coloring effects it can provide a great alternative for low spec computers, especially for office work with bright backgrounds and dark fonts, e.g., Excel or Word tasks.

While fully functional, further work is required to refine and improve the system. On the one hand, medical studies are required to evaluate the psychological impact of the system and therefore confirming its usefulness. Following this further studies are required to find optimal color profiles to maximize the effect. On the other hand, further technical improvements can be conducted. The support of more desktop environments could be realized and more clients should be implemented, allowing more specific use cases and optimal support for more kinds if needs. These clients could also use smart devices like watches or fitness tracers, allowing the integration of blood pressure into the emotion recognition.

Acknowledgments

We want to express our gratitude towards Andrea Steiner for her contribution towards the displayed mathematical content of this paper, providing guidance and corrections.

Further we want to thank Jakob Loskan for his advice concerning the technical details behind alpha blend and gamma shift, as well as the mathematics involved.

References

[1] L. Brodschelm, F. Gräber, D. Hieber, and M. Hermann, "Mood adaptive display coloring - utilizing modern machine learning techniques and intelligent coloring to influence the mood of pc users," in *ACHI 2021, The Fourteenth International Conference on Advances in Computer-Human Interactions*, 07 2021, pp. 48–54.

[2] F. Hawlitschek, L.-E. Jansen, E. Lux, T. Teubner, and C. Weinhardt, "Colors and trust: The influence of user interface design on trust and reciprocity," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, jan 2016, pp. 590–599.

[3] R. M. Rider, "Color psychology and graphic design applications." Liberty University, 2010.

[4] M. Kido, "Bio-psychological effects of color," *Journal of International Society of Life Information Science*, vol. 18, pp. 254–268, 2000.

[5] M. A. Maier, A. J. Elliot, and R. A. Barton, "Color in achievement contexts in humans," in *Handbook of Color Psychology*, A. J. Elliot, M. D. Fairchild, and A. Franklin, Eds. Cambridge University Press, pp. 568–584.

[6] R. A. Ainsworth, L. Simpson, and D. Cassell, "Effects of three colors in an office interior on mood and performance," *Perceptual and Motor Skills*, vol. 76, no. 1, pp. 235–241, Feb. 1993.

[7] I. Kondratova and I. Goldfarb, "Color your website: Use of colors on the web," in *Usability and Internationalization. Global and Local User Interfaces*, N. Aykin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 123–132.

[8] A. J. Elliot and M. A. Maier, "Color-in-context theory," in *Advances in Experimental Social Psychology*. Elsevier, 2012, pp. 61–125.

[9] C. Li and H. Shi, "Medical space oriented color psychology perception model," *Applied Mechanics and Materials*, vol. 587-589, pp. 461–467, 07 2014.

[10] K. R. Alexander and M. S. Shansky, "Influence of hue, value, and chroma on the perceived heaviness of colors," *Perception & Psychophysics*, vol. 19, no. 1, pp. 72–74, Jan. 1976.

[11] M. Rovai, "Real-time face recognition: An end-to-end project," 2018, last visited: 2021.06.14. [Online]. Available: https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348

[12] D. Yang, A. Alsadoon, P. Prasad, A. Singh, and A. Elchouemi, "An emotion recognition model based on facial recognition in virtual learning environment," *Procedia Computer Science*, vol. 125, pp. 2 – 10, 2018, the 6th International Conference on Smart Computing and Communications.

[13] S. Ghosh, E. Laksana, L.-P. Morency, and S. Scherer, "Representation Learning for Speech Emotion Recognition," *Interspeech 2016*, pp. 3603–3607, Sep. 2016.

[14] A. R. Smith, "Image compositing fundamentals," Tech. Rep., 1995.

[15] P. Berndt and other, "Pqiv git repository," 2021, last visited: 2022.05.24. [Online]. Available: https://github.com/phillipberndt/pqiv/

[16] L. C. Klopfenstein and other, "Ontopreplica git repository," 2021, last visited: 2022.05.24. [Online]. Available: https://github.com/LorenzCK/OnTopReplica

[17] (2021) The hudkit-projekt github page. Last visited: 2021.06.14. [Online]. Available: https://github.com/anko/hudkit

[18] J. de Vries, *Learn OpenGL - Learn Modern OpenGL Graphics Programming in a Step-by-step Fashion.* Kendall & Welling, 2020.

[19] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips, *Introduction to Computer Graphics*, 1st ed. Addison-Wesley, 1994.

[20] "Microsoft documentation - using gamma correction," 2021, last visited: 2021.10.25. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/direct3ddxgi/using-gamma-correction

[21] "Xorg - lib documentation," 2021, last visited: 2021.10.25. [Online]. Available: https://x.org/releases/current/doc/libX11/libX11/libX11.html#Introduction_to_Xlib

[22] "Freedesktop - xrandr," 2021, last visited: 2021.10.25. [Online]. Available: https://xorg.freedesktop.org/archive/current/doc/man/man1/xrandr.1.xhtml

[23] (2020, jul) Harvard health letter - blue light has a dark side. Last visited: 2021.06.14. [Online]. Available: https://www.health.harvard.edu/staying-healthy/blue-light-has-a-dark-side

[24] S. Mitropoulos, V. Tsiantos, A. Americanos, I. Sianoudis, and A. Skouroliakou, "Blue light reducing software applications for mobile phone screens: measurement of spectral characteristics and biological parameters." in *RAP 2019 Conference Proceedings*. Sievert Association, 2020, pp. 220–224.

[25] R. Sutherland, "What is in a color? the unique human health effects of blue light," in *Envirionmental Healt Perspectives volume 118*. News Focus, 2010, pp. 23–27.

[26] E. Joosten, G. Lankveld, and P. Spronck, "Colors and emotions in video games," *11th International Conference on Intelligent Games and Simulation, GAME-ON 2010*, 01 2010.

[27] R. Plutchik, "The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice," *American Scientist*, vol. 89, no. 4, pp. 344–350, 2001. [Online]. Available: http://www.jstor.org/stable/27857503

APPENDIX

SINGLE CHANNEL SHIFT GAMMA EQUATION

The color/γ-vector is only used for the ordered display of the color, while mathematically only the skalar product is used. Therefor, the vectors can be separated into its components and be compared pairwise. Equation (7) shows this for the red channel from Equations (1) and (2).
In the following Pixel is abbreviated with P and Taint with T.

$$\begin{aligned} & {P_R}^{\frac{1}{\gamma_R}} = P_R * (1-\alpha) + T_R * \alpha \\ \Leftrightarrow \quad & {P_R}^{\frac{1}{\gamma_R}} = P_R - P_R\alpha + T_R * \alpha \\ \Leftrightarrow \quad & {P_R}^{\frac{1}{\gamma_R}} = P_R + (T_R - P_R) * \alpha \\ \Leftrightarrow \quad & \frac{1}{\gamma_R} = log_{P_R}(P_R + (T_R - P_R) * \alpha) \\ \Leftrightarrow \quad & \gamma_R = \frac{1}{log_{P_R}(P_R + (T_R - P_R) * \alpha)} \end{aligned} \tag{7}$$

GAMMA SHIFT VECTOR EQUATION

This equation provides the basis for the vector based gamma shift. It can be derived by equating Equations (1) and (2).
In the following Pixel is abbreviated with P and Taint with T.

$$\begin{aligned} & \begin{pmatrix} {P_R}^{\frac{1}{\gamma_R}} \\ {P_G}^{\frac{1}{\gamma_G}} \\ {P_B}^{\frac{1}{\gamma_B}} \end{pmatrix} = \begin{pmatrix} P_R \\ P_G \\ P_B \end{pmatrix} * (1-\alpha) + \begin{pmatrix} T_R \\ T_G \\ T_B \end{pmatrix} * \alpha \\ \Rightarrow \quad & Applying\ Equation\ (7)\ for\ all\ 3\ channels \\ \Rightarrow \quad & \begin{pmatrix} \gamma_R \\ \gamma_G \\ \gamma_B \end{pmatrix} = \begin{pmatrix} \frac{1}{log_{P_R}(P_R + (T_R - P_R) * \alpha)} \\ \frac{1}{log_{P_G}(P_G + (T_G - P_G) * \alpha)} \\ \frac{1}{log_{P_B}(P_B + (T_B - P_B) * \alpha)} \end{pmatrix} \end{aligned} \tag{8}$$

Survey on Social Simulation and Knowledge Extraction from Simulation Results - Application for Constructing Life Planning Support Frameworks -

Takamasa Kikuchi
Graduate School of Business Administration
Keio University
Yokohama-city, Japan
e-mail: takamasa_kikuchi@keio.jp

Hiroshi Takahashi
Graduate School of Business Administration
Keio University
Yokohama-city, Japan
e-mail: htaka@keio.jp

***Abstract*—Asset formation for the retirement generation is a common issue around the world and has been widely discussed in various countries. We begin this paper by surveying the research on asset formation and life planning. Then, we show a data-driven life planning support framework based on social simulation. Based on the data and simulation results, this framework is intended to run simulations based on customer attribute data and to evaluate and validate measures for customers' retirement assets. The social simulation model is constructed based on finance theory. Machine learning methods are used for the analysis of customer features and evaluation of the policy measures. Moreover, the simulation results are represented by experience mapping techniques. The following are the key findings: our framework 1) allows for effective discussion of measures to avoid the depletion of retirement assets and 2) allows simulation results to be widely interpreted and shared not only by model developers and analysts but also by decision-makers and frontline personnel.**

Keywords- social simulation; knowledge extraction; financial retirement planning; experience mapping techniques

I. INTRODUCTION

In this paper, we first survey the research on asset formation and life planning. Then, we show a life planning support framework built based on data and social simulation. This paper extends our prior paper presented at eKNOW 2021 [1].

Asset formation for the retirement generation is a common issue around the world and has been widely discussed in various countries [1; 2]. As national, individual, and social measures, various measures are being discussed, such as raising the retirement age, establishing assets at a young age, and reducing spending. There has been little discussion, however, about asset withdrawal, and there is room to expand and modernize basic research and analysis to better consider current issues. To address this issue, it is important to have a simulation framework that enables effective discussion of measures to avoid depletion of retirement assets. In addition, it is desirable to have a method that allows simulation results to be widely interpreted and shared not only by model developers and analysts, but also by decision makers and those in charge in the field as shown in the survey section later.

Our framework is designed to run simulations based on data of customer attributes and to evaluate and validate measures for customers' retirement assets based on the data and simulation results. The social simulation model is built on finance theory [3]. Machine learning methods are also used for the analysis of customer characteristics and the evaluation of policy measures [4; 5]. Furthermore, we use experience mapping techniques to represent the simulation results extending prior research [6].

Here, this framework is intended to be used by financial planners and retail strategy planners who create life plans for their customers.

As a exemplification of the proposed framework, this paper presents a specific case study that focuses on customer asset formation and withdrawal for the retirement generation.

The structure of this paper is as follows: Section II introduces related work, Section III describes the proposed methodology, and Section IV explains the social simulation model we used. Section V provides applications, and in Section VI, we summarize the results.

II. RELATED WORK

In this section, we begin with a survey of asset formation and life planning (*A*). We begin by discussing portfolio selection theory among households, which is relevant to the issues addressed in this paper. Then, we examine surveys and research on asset formation and withdrawal in the postretirement period. Next, we introduce previous research on social simulation methods, the computer simulation methods used in this paper (*B*). There have been reports of research using social simulation to solve problems in social science. Finally, we mention the experience mapping techniques (*C*). The formal description of those techniques will be oriented in this paper for knowledge extraction from simulation results.

A. Asset Formation and Life Planning

1) Lifetime portfolio selection

Under several assumptions, the optimal ratio of risky asset holdings in the classical theory of household portfolio selection is determined by the expected rate of return on risky assets, the interest rate on safe assets, the variance of the rate of return on risky assets, and the relative risk aversion [7; 8]. On the other hand, it has been pointed out that many households do not hold any risky assets [9], suggesting the existence of mechanisms that cannot be explained by classical theory alone.

Bodie et al. [10] argued that the optimal risk asset ratio is high for young people with large human assets because they can cope with risks such as falling prices by increasing labor supply and working extra hours. Chen et al. [11] considered life insurance in addition to human asset allocation and investigated the optimal allocation of safe assets and life insurance when investment opportunities change over time.

On the other hand, empirical approaches suggest that households' investment in risky assets increases with age and then declines. Ameriks & Zeldes [12] showed that the allocation to equities over the life cycle follows a "mountainous" trend, using the United States (U.S.) as an example. This trend is common not only in the U.S. but also in other developed countries. In Japan, the stock allocation ratio peaks later than in the U.S. and Europe: 1) it reaches its maximum in the late 50s and 60s and 2) investment in stocks does not decrease substantially even after retirement. This implies that many Japanese households enter the stock market only after they get their retirement money [13].

2) Asset Formation and Withdrawal focused on Postretirement

Asset formation for the retirement generation is a common issue around the world [2] and has been widely discussed in various countries [14]. In the U.S., the empirical benchmark is a fixed withdrawal rate of 4% of initial assets [15]. On the contrary, some critics argue that a fixed withdrawal rate is inefficient [16] and that "rules" should be set to vary the withdrawal rate and amount [17; 18].

In Japan as well, various surveys, studies, and calculations have been conducted regarding the formation and withdrawal of assets after retirement [19; 20; 21]. The Financial System Council's estimate [20] calculated the required amount of funds to be withdrawn for an elderly couple and unemployed household in a simplified manner as follows:

Monthly net income and expenditure (−55,000 yen/month) × 30 years ≒ ⊿20 million yen

This estimation was easy to understand and drew national attention. On the other hand, it is based on the average household income and expenditure (deficit of approximately 55,000 yen per month) of an unemployed elderly couple whose only source of income is the pension and does not take into account each individual's income and expenditure situation and lifestyle. Other studies have focused on investment strategies in asset formation and estimated the probability of depletion using annual time series paths of asset prices by Monte Carlo simulation [21]. While this study takes into account the risky asset investment strategy as well as the macroenvironment (inflation rate), it does not explicitly address various individual characteristics. Furthermore, there is a study that simulates the future depletion rate of financial assets in real terms using macroeconomic data such as the amount of financial assets and disposable income of individuals [19]. While this study takes into account an individual's income and asset class, it does not address risky asset investment or inflation rates.

Kikuchi and Takahashi constructed a social simulation model that expresses asset formation and withdrawal before and after retirement [3]. They presented a simulation of the customer's asset situation at a future point in time, taking into account asset succession and price fluctuations of risky assets based on individual questionnaire data concerning asset formation and withdrawal before and after retirement [4; 5]. In this paper, we construct a life planning support framework based on the simulation model.

B. Social Simulation Method

A social simulation is an approach in the social sciences that uses computer simulation to analyze social phenomena [22; 23]. In recent years, many studies have been grounded in real data in the field of social simulation [24; 25].

Yamada et al. [24] proposed a method that utilizes actual data and agent simulation to solve problems in business and industry. They classified various types of airport behavior using real-world data, and they used agent-based simulation to successfully reproduce congestion conditions when new equipment was installed at Fukuoka Airport in Japan [25]. Many such analyses have also been reported, e.g., corporate behavioral analysis via modeling based on finance theory [26; 27].

Detailed analyses such as Yamada et al.'s can facilitate onsite decision-making and are expected to greatly contribute to efficient decision-making in both social and economic activities. Understanding and interpreting the model structure and simulation results, on the other hand, are not limited to model developers and analysts but may be conducted widely by decision-makers and field staff related to management and administration. As a result, having a methodology for extracting knowledge and insights from simulation log data as well as a framework for sharing the extracted knowledge and insights among stakeholders is critical.

C. Experience Mapping Techniques

In the design of products and services, "design thinking" has been attracting attention [28; 29; 30]. As observational methods, experience mapping techniques such as persona–scenario technique [31], customer journey map [32], empathy map, and so on [33] are used to uncover users' latent needs.

It is believed that the use of the experience mapping techniques has the effect of standardizing the perceptions among stakeholders, such as developers and marketers of products and services. This method may be an effective means of facilitating communication among stakeholders, which is a problem in the analysis of social simulation results, as discussed in the previous section.

1) Persona–scenario technique

In the field of marketing, human–computer interaction, and interaction design, "persona" is widely used as a method to support customer-oriented product and service design [6; 31; 34]. Persona marketing refers to the creation of an image of a person based on clear and specific data

about that real person. The use of personas leads to improving the quality of decision-making regarding the design of products and services.

Table I provides a sample of a persona comparison poster [34]. In this paper, referring to the previous literature [6], we analyze and write down the results of our social simulation in the form of the persona comparison poster.

TABLE I. EXAMPLE OF PERSONA COMPARISON POSTER [34]

Name	Tanner	Colbi	Austin	Preston
Age	9	7	12	3
Tagline	The tenacious tinkerer	The creative child	The active competitor	The precious preschooler
Personal Computer (PC) location	PC in family room only	Uses a PC in the family room and sometimes her brother's PC, when he lets her	Has a PC in his bedroom, rarely uses the PC in the family room	Uses the PC in the office with Mom
Internet Connection	Dial-up	Broadband	Broadband	Dial-up
PC/Internet Activity	Gaming, web surfing, some school work/re-search	Chatting with friends, surfing the web, school work/research, arts/crafts	Gaming, web surfing, tracking sports schedules, tracking favorite athletes, some school work/re-search	Educational games and light entertainment deemed worthy by Mom

2) Other experience mapping techniques

Other experience mapping techniques include empathy map, customer journey map, service blueprint, and so on [33] (Fig. 1). The customer journey map [32] is the most basic and widely used visual description method in service design, and it is used in this paper. We write down the results of our social simulation in the form of a customer journey map, as we did with the previous persona–scenario technique.

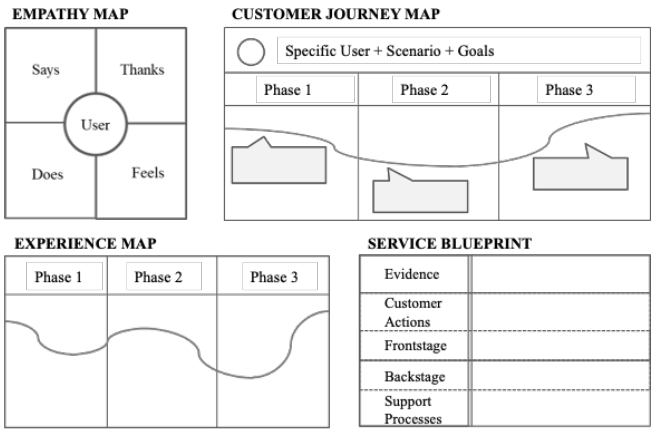


Figure 1. Various experience mapping techniques [33]

III. METHODOLOGY

In this section, we outline our proposed life planning support framework for retirees based on data and social simulation. This framework is based on our previous studies [3; 4; 5; 6].

A. Outline

Our framework is designed to run simulations based on real data of customer attributes and to evaluate measures for customers' retirement assets based on the data and simulation results. Fig. 2 depicts an overview of the proposed framework, which corresponds to the concept of cyber-physical systems. Our framework consists of the following two procedures:

1) Data augmentation by social simulation: Real data and simulation logs/paths are integrated and treated as augmented customer attribute data (See I in Fig. 2).

2) Knowledge extraction: From the above-augmented data, knowledge about the sustainability of assets for each customer is extracted using machine learning methods and experience mapping techniques (See II in Fig. 2).

In detail, the real data is a large survey data containing various attributes of customers (Section V-A). In this paper, we use cross-sectional data from individual questionnaires. Then, the social simulation model is constructed based on finance theory [36; 37] (Section IV). The model addresses asset formation in retirement, taking into account asset succession and risky asset price fluctuations. Based on the model, simulations are run to perform what-if analysis of the customer's asset's sustainability (Section V-D). The proportion of asset depletion in future, as obtained from the simulation results, is treated as augmented data for the original questionnaire data. Moreover, machine learning methods [38; 39] are used for the analysis of customer features (Section V-C) and evaluation of the policy measures (Section V-E). Finally, the results are represented using experience mapping techniques [34; 32] (Section V-F and G).

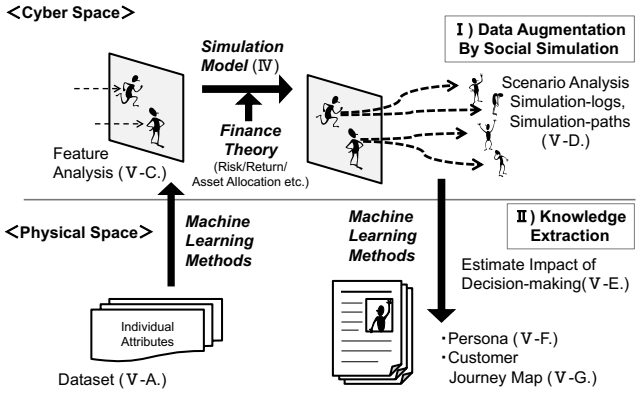


Figure 2. Basic Architecture of the proposed Life Planning Support framework

Here, this framework is intended to be used by financial planners and retail strategy planners who create life plans for their customers.

B. The Usefulness of the Proposed Framework

a) Perspectives on the advanced analysis of the sustainability of personal assets:

Regarding asset depletion, many studies have analyzed macrostatistical data, e.g., the amount of financial assets and disposable income [19]. Furthermore, based on market data, the effect of price fluctuations on owned assets and asset depletion has been examined [21]. However, previous studies only used actual data to express some of the characteristics of individuals. In short, traditional analysis has frequently focused on people with specific characteristics to simulate asset depletion scenarios. Additionally, in those studies, ad hoc analysis was required for each person to examine possible measures to be taken.

On the other hand, the proposed framework is capable of handling various attributes of individuals. Individual investment preferences and the amount of assets they will inherit in the future, for example, have rarely been addressed in previous studies. In addition, our framework can comprehensively and semiautomatically specify potential life planning measures for each customer. The framework will allow for effective discussion of measures to avoid the depletion of retirement assets.

b) Perspectives on visualization of simulation results

As mentioned in Section II-B, it is important to have a methodology for extracting knowledge and insights from simulation log data and a framework for sharing the extracted knowledge and insights among stakeholders.

In the proposed method, simulation paths and logs, which are simulation results, are classified using machine learning methods and then visualized using experience mapping methods. By describing the results using those methods, a) the results can be compared in a graphical format, and b) the perspectives of customers and users can be expressed. This will allow the results to be widely interpreted and shared, not just by model developers and analysts, but also by decision-makers and frontline personnel. Furthermore, it has the potential to improve communication among stakeholders during the analysis and sharing of simulation results.

C. Limitation of the Proposed Framework

One of the limitations of this analysis is that there is arbitrariness on the part of the modeler as to which attributes of the targeted individuals are reflected in the simulation model. Furthermore, in the process of extracting knowledge from simulation results, the analyst's discretion in selecting which results to focus on and describe formally may exist. Of course, the social simulation model used must be an accurate representation of the real world.

IV. Social Simulation Model

In this section, we describe the social simulation model used in this paper. This model is based on our previous studies [3; 4; 5].

A. Outline

As an example of the proposed framework, we show a simulation of the customer's asset situation at a future point in time, taking into account asset succession and price fluctuations of risky assets based on individual questionnaire data concerning asset formation and withdrawal in old age.

We constructed a computer simulation model that expresses asset formation and withdrawal before and after retirement (Fig. 3).

Each actor in the model has a specific asset balance at a certain age. According to the actor's status, each actor has both regular income and expenditure (cash inflow and outflow) and unexpected income and expenditure (depending on life events) (before and after retirement). Each actor's characteristics are expressed statistically. In addition, by manipulating the attributes of the actors, what-if analysis can be performed to examine responses to the implementation of a particular policy or decision-making.

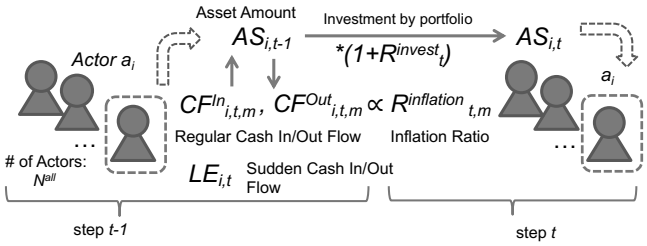


Figure 3. Conceptual diagram of a simulation model

The assets held by each actor include cash, deposits, and risk assets. Risk assets are fully invested in a portfolio of traditional assets and provide returns according to the risk of the portfolio. Furthermore, regular income and expenditure are adjusted to account for inflation. The external environment is comprised of the portfolio's risk–return profile, the inflation rate, and the variance of each, as described further below.

B. Actor

Let A be the set of actors and let $\#A = N^{all}$. Actor a_i has the following attributes in step t of the simulation and inflation scenario m: age $age_{i,t}$, retirement age $age_i^{retired}$, cash and deposit balance $CA_{i,t}$, risk asset balance $RA_{i,t}$, cash inflow $CF^{In}_{i,t,m}$, cash outflow $CF^{Out}_{i,t,m}$, cash flow from life event $LE_{i,t}$, and total asset balance $AS_{i,t} = CA_{i,t} + RA_{i,t}$.

$$A = \{a_i = (i, age_{i,t}, age_i^{retired}, CA_{i,t}, RA_{i,t}, CF^{In}_{i,t,m}, CF^{Out}_{i,t,m}, AS_{i,t}, LE_{i,t})\}$$

Here, age is expressed as follows.

$$age_i \in \{age_{i,0}, age_{i,0} + 1, \ldots, age_i^{retired}, \ldots\}$$

Then, the simulation time step, a single step represents one year in real-time.

C. External Environment

The return and inflation rate of portfolio *j* are generated in time series by Monte Carlo simulation as follows, where the number of trials is *K*.

The portfolio return (annual) is expressed as

$$R^{invest}{}_t = X_{1,t}\,\sigma + \mu.$$

The inflation rate (annual) is given as

$$R^{inflation}{}_{t,m} = (\rho X_{1,t} + \sqrt{(1-\rho^2)}\,X_{2,t})\,\sigma^{inflation} + \mu^{inflation}{}_m.$$

Here, σ is the risk of portfolio, μ is the expected return rate of portfolio, $\sigma^{inflation}$ is the standard deviation of the inflation rate, $\mu^{inflation}{}_m$ is the expected inflation rate at scenario m, ρ is a correlation coefficient between portfolio and the inflation rate. $X_1, X_2 \sim N(0,1)$, and $cov[X_1, X_2] = 0$.

The cumulative value *IRC* of the inflation rate referred to as cumulative inflation rate, is expressed as

$$IRC_{t,m} = IRC_{t-1,m}(1 + R^{inflation}{}_{t,m}),\ IRC_{0,m} = 1.$$

D. Cash Inflow/Outflow

Retirement cash inflows and outflows *CF* in step *t* are determined by considering the asset class to which actor *i* belongs and the cumulative inflation rate as follows:

$$CF^{In}{}_{i,t,m} = CF^{In}{}_{i,0,m}\,(1 + IRC_{t,m}),$$
$$CF^{Out}{}_{i,t,m} = CF^{Out}{}_{i,0,m}\,(1 + IRC_{t,m})$$

To keep the estimated possibility of withdrawal conservative, the net cash flow before retirement is set to zero.

E. Asset Formation and Withdrawal Rules

The cash and deposit balance and risk asset balance in each simulation step are varied according to the following rules. This expresses the preferential withdrawal of highly liquid cash and deposits at the asset withdrawal stage.

if $CA_{i,t} + CF^{In}{}_{i,t,m} - CF^{Out}{}_{i,t,m} >= 0$

then

$$CA_{i,t+1} = CA_{i,t} + CF^{In}{}_{i,t,m} - CF^{Out}{}_{i,t,m} + LE_{i,t}$$
$$RA_{i,t+1} = RA_{i,t}\,(1 + R^{invest}{}_t)$$

else

$$CA_{i,t+1} = CA_{i,t} + LE_{i,t}$$
$$RA_{i,t+1} = RA_{i,t}\,(1 + R^{invest}{}_t) + CF^{In}{}_{i,t,m} - CF^{Out}{}_{i,t,m}$$

F. Asset Depletion Rate

For the *K* trials, the number of times the asset balance becomes negative at age τ and inflation scenario m is denoted $K^{shortage}$, and the asset depletion rate, hereafter referred to as the depletion rate, is expressed as

$$R_{i,m,\tau}{}^{shortage} = K_{i,m,\tau}{}^{shortage}/K.$$

V. APPLICATIONS

In this section, we show applications of our framework.

A. Dataset: Individual Attributes

We use the individual questionnaire data from the "Awareness Survey on Life in Old Age for Before and After Retirement Generations" conducted by the MUFG Financial Education Institute [40]. The survey was aimed at men and women aged 50 and up. The survey area was Japan, and there were 6,192 valid responses. This questionnaire thoroughly investigated each individual's asset status (current asset balance and expected income/expenditure in old age), planned asset inheritance amount, investment stance, and outlook for old age, and so on.

The basic statistics of the questionnaire on age, current asset balance, asset balance to be inherited, and percentage of risky assets held are shown in Table II [5].

TABLE II. BASIC STATISTICS OF QUESTIONNAIRE (abstract)

Statistics	Question matters (extract)			
	Age	Asset Balance Current	Asset Balance to be Inherited	Risk Asset Holding Ratio
Mode	70	30-50 m yen	0 m yen	0%
Median	64.5	15-20 m yen	0 m yen	0%
Max	91	100- m yen	100- m yen	90%-
Min	50	0-1 m yen	0 m yen	0%
First Quartile	57	7-8 m yen	0 m yen	0%
Third Quartile	71	30-50 m yen	0 m yen	20-30%
# of Samples	6,192	6,192	5,342	6,068

B. Model Implementation and Simulation Environment

The social simulation model presented in Section IV was implemented using the python language; the versions of python and jupyter notebook are 3.8.5 and 6.1.4, respectively. The simulation environment was MacBook Air (13-inch, 2017), processor: Intel Core i7 2.2GHz dual core, memory: 8GB 1600 MHz DDR3, OS: macOS Big Sur ver 11.4.

C. Feature Analysis of Individual Questionnaire Data

Based on the above individual questionnaire data, we established several patterns of "possible person attributes" through segmentation by feature analysis [4; 5].

TABLE III. QUESTIONNAIRE ITEMS USED FOR FEATURE ANALYSIS

Item	Question matters
Attributes	Age, Sex, Household composition, etc.
Financial Statue	Stock data: Asset Balance(Current), Asset Balance(to be Inherited), etc.
	Flow data: Regular Cash In/Out Flow, etc.
Risk Preference	Investment Experience, Risk Asset Holding Ratio, etc.

For the individual data, the following items were targeted (Table III), and clustering was performed using the k-means method [38]. The number of clusters was set to five in this

case based on the results of the elbow chart and silhouette analysis, which are frequently used to determine the number of clusters. There were 4,592 samples available for all items in the data.

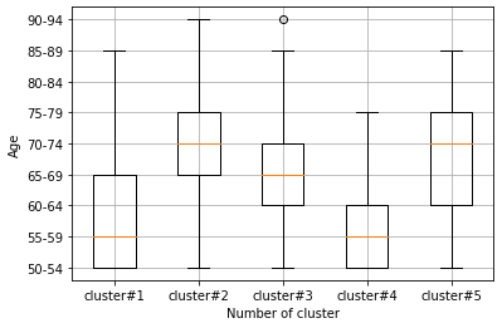


Figure 4. (a) Distribution of Age Groups for Each Cluster

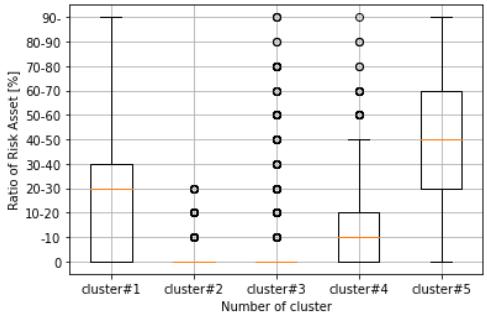


Figure 4. (b) Distribution of Risk Assets Holding Ratio for Each Cluster

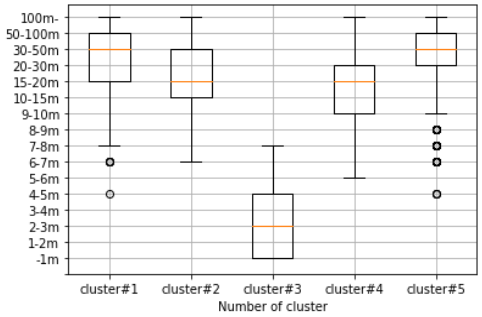


Figure 4. (c) Distribution of Current Financial Asset Balances for Each Cluster

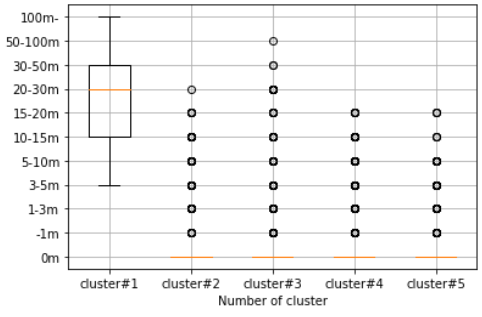


Figure 4. (d) Distribution of Financial Assets to be Inherited for Each Cluster

From the obtained clustering results (clusters #1–#5), Figs. 4(a)–4(d) show the distribution of the answers to typical questionnaire items for each cluster as a box plot.

The median age group *age* was 55–59 years for clusters #1 and #4, 65–69 years for cluster #3, and 70–74 years for clusters #2 and #5, as shown in Fig. 4(a).

The median holding ratio of risk assets R^{risk} was 0% for clusters #2 and #3, 0%–10% for cluster #4, 20%–30% for cluster #1, and 40%–50% for cluster #5, as shown in Fig. 4(b).

The median current balances of financial assets FA^{now} was 2–3 million yen for cluster #3, 15–20 million yen for clusters #2 and #4, and 30–50 million yen for clusters #1 and #5, as shown in Fig. 4(c).

The median balance of financial assets to be succeeded FA^{future} was zero for clusters #2–#5 and 20–30 million yen for cluster #1, as shown in Fig. 4(d).

Risk asset holding ratio, i.e., investment preferences of individuals and the amount of assets they will inherit in the future have rarely been addressed in previous studies. So, our proposed framework is capable of handling various attributes of individuals.

D. *Asset Formation Simulation: Depletion Rate Based on Individual Questionnaire Data*

Using the social simulation model referred to in Section IV, we performed computer simulations of asset formation and withdrawal based on the cluster set described in the previous section (Table IV). Then, we estimated the asset depletion status of representative people who were typified by the individual questionnaire data [4; 5].

Here, the annual income and expenditure CF^{net} for each asset class was set from macro statistics data [41] according to the current balance of financial assets. The correspondence with the model described in Section IV is expressed as follows.

$N^{all} = 5$ (cluster#1–#5), $CA_{i,0} = FA^{now}{}_i * (1 - R^{risk}{}_i)$, $RA_{i,0} = FA^{now}{}_i * R^{risk}{}_i$, $CF^{Out}{}_{i,0,m} - CF^{In}{}_{i,0,m} = CF^{net}{}_i$

TABLE IV. Setting Attributes for Each Cluster

# of cluster	Attributes			
	age	FA^{now}	FA^{future}	R^{risk}
#4	57	17.5 m yen	none	5%
#1	57	40.0 m yen	25.0 m yen	25%
#3	67	2.5 m yen	none	0%
#2	72	17.5 m yen	none	0%
#5	72	40.0 m yen	none	45%

We also define the parameters as follows: $age_i^{retired} = 60$, the actor's age at which a life event occurs: $age_{i,t'} = 70$, $LE_{i,t'} = FA^{future}{}_i * R^{future}$. Here R^{future} is the ratio of asset succession (representing the ratio of the actual balance of financial assets to be succeeded). And t' is the age at which

a life event (asset succession) occurs. In this paper, the above parameters are called "Case of Making Basic Decisions."

Other parameter settings, e.g., the portfolio's risk–return profile and inflation rate, are shown in Table V. Note that the risk–return of the portfolio was set assuming a portfolio comprising foreign stocks and bonds [3]. The expected inflation rates were according to three patterns, i.e., (1) no inflation (0%), (2) moderate inflation (actual results for the past 30 years in Japan [42]: 0.53%), and (3) 2% inflation (monetary easing target). Here, the standard deviation of the inflation rate was the same as pattern (2), which is the actual result for the past 30 years in Japan.

TABLE V. Parameter Settings: Case of Making Basic Decisions

Item	Value
Curbing of Expenditure	Without
$age^{retired}$	60
R^{future}	100%
μ_j, σj	(6.37%, 18.0%)
$\mu_{inflation}$	{ 0.0%, 0.53%, 2.0%}
$\sigma_{inflation}$	1.26%
K	10,000

The depletion rate at age 90 and age 100 by cluster and inflation scenario is shown in Table VI [4].

TABLE VI. SIMULATION RESULT: DEPLETION RATES IN CASE OF MAKING BASIC DECISIONS

# of cluster	Depletion rates by inflation scenario					
	(1) No inflation		(2) Moderate inflation		(3) 2% inflation	
	Age: 90 (%)	Age: 100 (%)	Age: 90 (%)	Age: 100 (%)	Age: 90 (%)	Age: 100 (%)
#4	34	75	60	86	93	98
#1	0	0	0	0	0	0
#3	100	100	100	100	100	100
#2	0	34	0	94	0	100
#5	0	0	0	1	0	5

The depletion rate of cluster #4 increases according to the high inflation scenario, and the depletion rate of cluster #1 is zero in all scenarios. However, keep in mind that the simulation was based on the assumption that financial assets are inherited expectedly. In all scenarios, the depletion rate of cluster #3 was 100%. Cluster #2, similar to cluster #4, exhibits a high depletion rate in a high inflation scenario. Asset depletion was observed with a low probability in the limited case of high inflation at the age of 100 for cluster #5.

E. Estimate of Impact of Various Decisions on Depletion Rates

We conducted a what-if analysis in which the actors made various decisions to control asset depletion. By calculating feature importance using machine learning methods, we examined the effectiveness of these decisions [4].

We analyzed decisions that have a large effect on the depletion rate for the clusters set shown in Table IV. We considered the following decisions: 1) portfolio's risk–return profile, 2) retirement age, 3) curbing of expenditure, and 4) asset succession.

- *What-if analysis and calculating feature importance*

The parameter settings are shown in Table VII. Here, the assumed decision-making patterns are as follows. The annual income and expenditure CF^{net} by asset class has two patterns, i.e., with and without curbing of expenditure. The retirement ages are 60, 65, and 70 years. There are three asset succession patterns, i.e., 100%, 50%, and 0%, and four portfolio risk setting patterns, i.e., 18%, 12%, 6%, and 0% (returns are set according to the corresponding figures, i.e., 6.37%, 4.68% 2.87%, and 0.01%; see reference [3]). For each cluster set, 72 decision-making patterns were generated. Note that the other parameters were the same as those shown in Table V. In this paper, the above parameters are called "Case of Making Various Decisions."

TABLE VII. Parameter Settings: Case of Making Various Decisions

Item	Value
Curbing of Expenditure	{Without, With}
$age^{retired}$	{60, 65, 70}
R^{future}	{100%, 50%, 0%}
μ_j, σj	{(6.37%, 18.0%), (4.68%, 12.0%), (2.87%, 6.0%), (0.01%, 0.0%)}
$\mu_{inflation}$	{ 0.0%, 0.53%, 2.0%}
$\sigma_{inflation}$	1.26%
K	10,000

Here, we focus on the status of asset depletion and nondepletion at the specific age for each cluster. The importance of variables that classify depletion and nondepletion for each cluster was calculated using the random forest method [39] (Fig. 5).

For cluster #4, the importance of the portfolio risk setting was relatively high. Cluster #1 showed the highest importance in order of asset succession ratio, portfolio risk settings, and curbing of expenditure. For clusters #3 and #2, the importance of curbing expenditure was extremely high. In addition, cluster #5 had the highest importance in order of portfolio risk setting and curbing of expenditure.

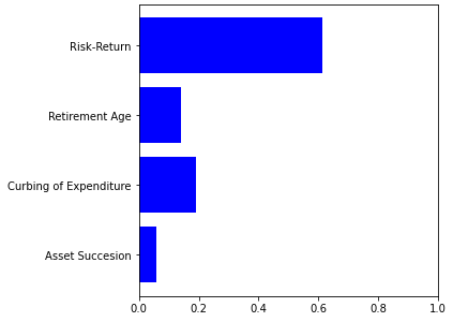


Figure 5. (a) Variable Importance for Each Cluster (#4)

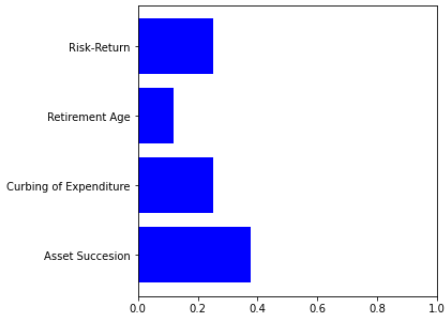


Figure 5. (b) Variable Importance for Each Cluster (#1)

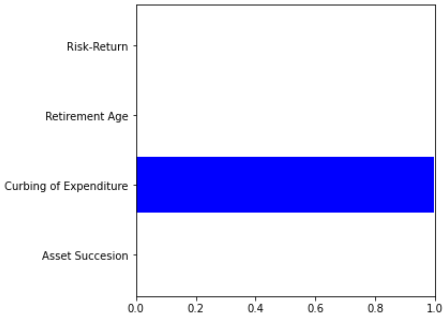


Figure 5. (c) Variable Importance for Each Cluster (#3)

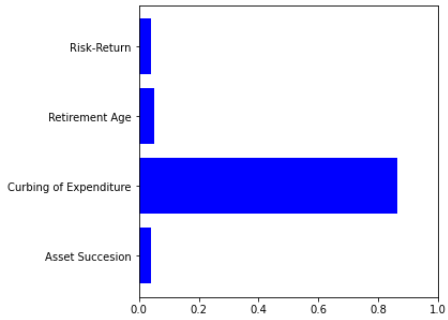


Figure 5. (d) Variable Importance for Each Cluster (#2)

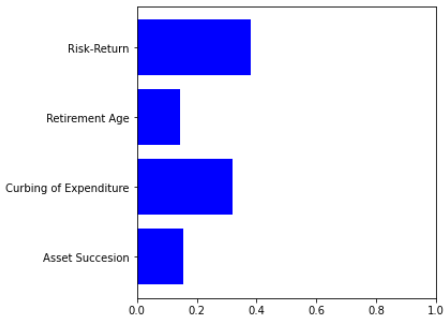


Figure 5. (e) Variable Importance for Each Cluster (#5)

- *Possible Effective Decisions for Each Cluster*

Next, from the results shown in Table VII and Fig. 5, we considered measures each individual can take to reduce the depletion rate (Table VIII).

Cluster #4: High depletion rate in the high inflation scenario. Appropriate risk-taking for inflation hedging and increasing retirement age could be effective measures [Fig. 5(a)].

Cluster #1: The depletion rate is low in all scenarios. However, this simulation assumed that financial assets are inherited as expected. Regarding variable importance, the ratio of asset succession was the highest [Fig. 5(b)], and appropriate and steady asset succession is important.

Cluster #3: The depletion rate was extremely high in all scenarios. Note that curbing expenditure was the only option among the decisions compared in this paper [Fig. 5(c)]. For cluster #3, drastic measures are required, e.g., curbing expenditure and expanding social security.

Cluster #2: The depletion rate was high in the moderate and high inflation scenario at age 100. Here, curbing expenditure is considered an effective action [Fig. 5(d)].

Cluster #5: Here, the depletion rate was high in a limited scenario. As a countermeasure, it is conceivable to take appropriate risks [Fig. 5(e)]. This cluster showed a high proportion of risk assets (Table IV), and it is important to avoid excessive risk to prevent price fluctuations (decreases) of the held risk assets.

Table VIII summarizes examples of actions each cluster could take to reduce the depletion rate. In addition to the results of the previous section (Table VI), our framework is capable of comprehensively and semiautomatically specifying possible life planning measures for each customer [4].

TABLE VIII. ASSUMED COUNTERMEASURES FOR EACH CLUSTER

# of cluster	Countermeasures (example)
#4	Appropriate risk taking for inflation hedging, increase retirement age
#1	Appropriate and steady asset succession
#3	Curbing expenditure, expanding social security
#2	Curbing expenditure
#5	Avoid excessive risk to prevent price fluctuations

F. Knowledge Extraction Using Experience Mapping Techniques: Persona–scenario Technique

From this section onward, the above simulation results are formally described using the experience mapping techniques. First, we use the persona–scenario technique [6; 34].

Table IX describes the contents of Tables IV, VI, and VIII. in the form of a persona comparison poster. Here, we exemplify Table IX (a) showing the moderate inflation scenario and Table IX (b) showing the 2% inflation scenario.

From the original questionnaire, attributes "Age group," "Current balances of financial assets," "Holding ratio of risk assets," and "Financial assets to be succeeded" were selected and entered as the attributes obtained by feature analysis. Furthermore, the statuses "Depletion rate at age 90" and "Depletion rate at age 100" were chosen and entered from the simulation results. In addition, effective measures to reduce the rate of depletion are described for each cluster.

The benefit of using the persona–scenario method for formal description is that it allows you to see and compare the differences in attributes and states between clusters for each simulation scenario. This limits the ability of stakeholders other than modelers and analysts to interpret simulation results.

TABLE IX. (a) PERSONA COMPARISON POSTER: MODERATE INFLATION SCENARIO

Cluster #	#4	#1	#3	#2	#5
Age group	55–59	55–59	65–69	70–74	70–74
Current balances of financial assets	15–20 m yen	30–50 m yen	2–3 m yen	15–20 m yen	30–50 m yen
Holding ratio of risk assets	0%–10%	20%–30%	0%	0%	40%–50%
Financial assets to be succeeded	0 m yen	20–30 m yen	0 m yen	0 m yen	0 m yen
Depletion rate at age 90	60%	0%	100%	0%	0%
Depletion rate at age 100	86%	0%	100%	94%	1%

TABLE IX. (b) PERSONA COMPARISON POSTER: 2% INFLATION SCENARIO

Cluster #	#4	#1	#3	#2	#5
Age group	55–59	55–59	65–69	70–74	70–74
Current balances of financial assets	15–20 m yen	30–50 m yen	2–3 m yen	15–20 m yen	30–50 m yen
Holding ratio of risk assets	0%–10%	20%–30%	0%	0%	40%–50%
Financial assets to be succeeded	0 m yen	20–30 m yen	0 m yen	0 m yen	0 m yen
Depletion rate at age 90	93%	0%	100%	0%	0%
Depletion rate at age 100	98%	0%	100%	100%	5%

G. Knowledge Extraction Using Experience Mapping Techniques: Customer Journey Map

This section provides a formal description of the simulation results using the customer journey map. Figure 6 describes the contents of Tables IV, VI, and VIII in the form of the customer journey map [32]. Here, the cases of both clusters # 4 and # 1 are illustrated.

The attributes obtained by feature analysis from the original questionnaire are described in the "User profile" section. In addition, the "Scenario and Goal" section describes simulation scenarios and results (future state for each cluster). Furthermore, the entire period was divided into four phases based on the state of the actors in the simulation. Following that, we included a summary of the changes in the balance of assets held as well as the status of actors in each phase.

The state of the actor is described in the balloon in the figure and is based on the simulation result. Discussions between stakeholders, on the other hand, can be used to describe the thoughts and feelings of expected customers and users. Thus, the formal description of simulation results using a customer journey map has the advantage of allowing the customer or user's perspective to be expressed as well.

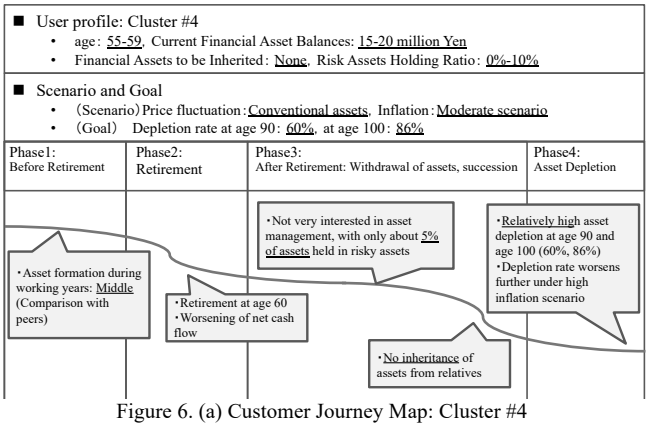


Figure 6. (a) Customer Journey Map: Cluster #4

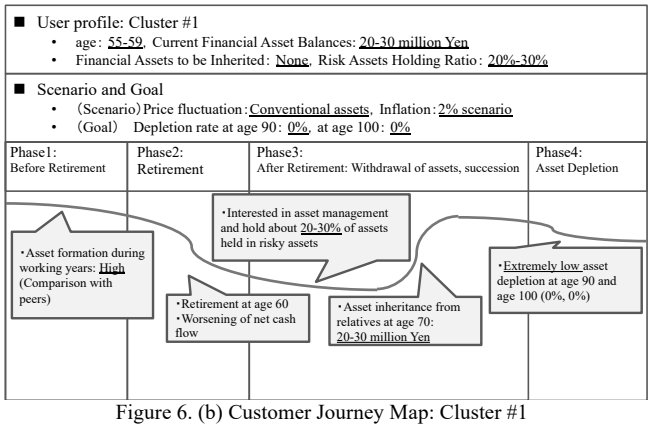


Figure 6. (b) Customer Journey Map: Cluster #1

VI. Concluding Remarks

In this paper, we first surveyed the research on asset formation and life planning. Then, we showed a life planning support framework built based on data and social simulation.

This framework was designed to run simulations based on data of customer attributes and to evaluate and validate measures for customers' retirement assets based on the data and simulation results. The social simulation model was built using finance theory. Machine learning is also used for customer feature analysis (k-means method) and policy measure evaluation (random forest method). Furthermore, the simulation results were represented using experience mapping techniques (persona-scenario technique and customer journey map).

As a exemplification of the proposed framework, we showed a specific case study that focuses on customer asset formation and withdrawal for the retirement generation. As the data source of customer attributes, we used large-scale individual questionnaire data.

The main findings are as follows: 1) our framework could effectively discuss measures to avoid the depletion of retirement assets. In addition, our framework is capable of dealing with a wide range of individual characteristics (Fig. 4) and specifying comprehensively and semiautomatically possible life planning measures for each customer (Tables VI and VIII). 2) Using our framework, simulation results can be widely interpreted and shared not only by model developers and analysts but also by decision-makers and the frontline personnel. By describing the simulation results using experience mapping techniques, a) the results could be compared in an overview (Table IX), and b) the viewpoints of customers and users can be expressed (Fig. 6).

Limitation of the proposed framework are: 1) there is arbitrariness on the part of the modeler as to which attributes of the targeted individuals are reflected in the simulation model, 2) in the process of extracting knowledge from simulation results, the analyst's discretion in selecting which results to focus on and describe formally may exist, 3) the social simulation model used must be an accurate representation of the real world.

By using the proposal system, financial planners and retail strategic planners could obtain knowledge feedback directly related to life planning. It is expected that financial institutions will also be able to provide detailed advice and counsel.

Future work would be to examine formal description methods for interpreting and sharing simulation results.

ACKNOWLEDGMENTS

This work is supported in part by the Grant of Foundation for the Fusion Of Science and Technology. The authors would like to thank Enago (www.enago.jp) for the English language proof.

References

[1] Kikuchi, T. and Takahashi, H., "Survey and Application: Constructing Life Planning Support System for Retirement Planning using Social Simulation," The Thirteenth International Conference on Information, Process, and Knowledge Management (eKNOW 2021), In proc., pp. 27-28, 2021.

[2] Investing in (and for) Our Future, World Economic Forum White Paper (2019) http://www3.weforum.org/docs/WEF_Investing_in_our_Future_report_2019.pdf, last accessed 2022/5/29.

[3] Kikuchi, T. and Takahashi, H., "Policy Simulation on Retirement Planning Considering Individual Attributes," Journal of the Japan Society for Management Information, Research Note, vol. 30, no. 2, pp. 105-119, 2021. (in Japanese)

[4] Kikuchi, T. and Takahashi, H., "Policy Simulation for Retirement Planning Based on Clusters Generated from Questionnaire Data," In: Jezic G., Chen-Burger J., Kusek M., Sperka R., Howlett R.J., Jain L.C. (eds) Agents and Multi-Agent Systems: Technologies and Applications 2021. Smart Innovation, Systems and Technologies, Springer, Singapore, vol. 241, pp. 285-298, 2021.

[5] Kikuchi, T. and Takahashi, H., "Life Planning Support System for Older Generations using Social Simulation Log Analysis," Transactions of the Society of Instrument and Control Engineers, vol. 57, issue 12, pp. 552-562, 2021. (in Japanese)

[6] Kikuchi, T. and Takahashi, H., "A Persona Design Method Based on Data Augmentation by Social Simulation," The IEEE/ACIS 21st International Fall Conference on Computer and Information Science (ICIS 2021-Fall), In proc., 2021.

[7] Merton, R. C., "Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case," Review of Economics and Statistics, vol. 51, no. 3, pp. 247-257, 1969.

[8] Samuelson, P. A., "Lifetime Portfolio Selection by Dynamic Stochastic Programming," Review of Economics and Statistics, vol. 51, no. 3, pp. 239-246, 1969.

[9] Mankiw, N. G. and Zeldes, S. P., "The Consumption of Stockholders and Nonstockholders," Journal of Financial Economics, vol. 29, no. 1, pp. 97-112, 1991.

[10] Bodie, Z., Merton, R. C. and Samuelson, W., "Labor Supply Flexibility and Portfolio Choice in a Life-Cycle Model," Journal of Economic Dynamics and Control, vol. 16, no. 3-4, pp. 427-449, 1992.

[11] Chen, P., Ibbotson, R. G., Milevsky, M. A. and Zhu, K. X., "Human Capital, Asset Allocation, and Life Insurance," Financial Analysts Journal, January/February, 2006.

[12] Ameriks, J. and Zeldes, S. P., "How Do Household Portfolio Shares Vary with Age?," TIAA-CREF Institute, TIAA-CREF Working Paper, 2004.

[13] Iwaisako, T., "Household asset allocation," Securities analysts journal, vol. 44, no. 8, pp. 6–14, 2006. (in Japanese)

[14] Fujibayashi, H., "Individual assetmanagement and retirement income securing-life cyclemodel and asset reversal strategy," Securities analysts journal, vol. 52, no. 10, pp. 50–55, 2014. (in Japanese)

[15] Bengan, W. P., "Determining Withdrawal Rates Using Historical Data," Journal of Financial Planning, pp. 767-777, 1994.

[16] Scott, J.S., Sharpe, W.F. and Watson, J.G., "The 4% Rule – At What Price ?," Journal of Investment Management, Third Quarter, 2008.

[17] Guyton, W.T. and Klinger, W., "Decision rules and maximum initial withdrawal rates," Journal of Financial Planning, vol. 19, article 6, 2006

[18] Spitzer, J.J., "Retirement withdrawals: an analysis of the benefits of periodic "midcourse" adjustments," Financial Services Review, vol. 17, pp. 17-29, 2008.

[19] Yokoyama, et al., "Future forecast/policy simulation analysis on private asset formation," MURC report, 2018. (in Japanese). https://www.murc.jp/report/rc/policy_rearch/politics/seiken_180112_2/, last accessed 2022/5/29.

[20] The Financial System Council (2019) (in Japanese) https://www.fsa.go.jp/singi/singi_kinyu/tosin/20190603/01.pdf, last accessed 2022/5/29.

[21] Kato, Y., "Post-retirement asset management framework," Securities analysts journal, vol. 56, no. 8, pp. 19–28, 2018. (in Japanese)

[22] Gilbert, N. and Doran, J., (eds.) "Simulating Societies: The Computer Simulation of Social Phenomena," University College of London Press, 1994.

[23] Carley, K. M. and Prietula, J. (eds.) "Computational Organization Theory," Lawrence-Erlbaum, 1994.

[24] Yamada, H., Ohori, K., Iwao, T., Kira, A., Kamiyama, N., Yoshida, H. and Anai, H., "Modeling and managing airport passenger flow under uncertainty: A case of Fukuoka Airport in Japan," 9th International Conference on Social Informatics (SocInfo), LNCS 10540, pp. 419-430, 2017.

[25] Ohori, K., "Systems Science Approaches Toward Social Implementation of AI," Journal of the Japanese Society for Artifical Intelligence, vol. 35, no. 4, pp. 542-548, 2020. (in Japanese)

[26] Takahashi, H., Takahashi, S. and Terano, T., "Analyzing the validity of passive investment strategies employing fundamental indices through agent-based simulation," In KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, Springer, Berlin, Heidelberg, pp. 180-189, 2011.

[27] Kikuchi, T., Kunigami, M., Yamada, T., Takahashi, H. and Terano, T., "Agent-based Simulation of Financial Institution's Investment Strategy under Easing Monetary Policy on Operative Collapses," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 22, no. 7, pp. 1026-1036, 2018.

[28] Kelley, T. and Litterman, J., "The art of innovation: Lessons in creativity from IDEO, America's leading design firm," Doubleday, 2001.

[29] Moggridge, B., "Designing Interactions," The MIT Press, 2006.

[30] Brown, T., "Design Thinking," Harvard Business Review, June, 2008.

[31] Cooper, A., "The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity," Macmillan, 1999.

[32] Kalbach, J., "Mapping Experiences: A Complete Guide to Creating Value through Journeys, Blueprints, and Diagrams," 1st edn, O'Reilly Media, 2016.

[33] Gibbons, S., "UX Mapping Methods Compared: A Cheat Sheet," Nielsen Norman Group, 2017. https://www.nngroup.com/articles/ux-mapping-cheat-sheet/, last accessed 2022/1/7.

[34] Pruitt, J. and Adlin, T. "The Persona Lifecycle: Keeping People in Mind Throughout Product Design (Interactive Technologies)," Morgan Kaufmann, 2006.

[35] Goodwin, K., "Designing for the Digital Age: How to Create Human-Centered Products and Services," Wiley, 2009.

[36] Ingersoll, J. E., "Theory of Financial Decision Making," Row- man & Littlefield Publishers, 1987.

[37] Duffle, D., "Dynamic Asset Pricing Theory," Princeton University Press, 2001.

[38] MacQueen, J., "Some Methods for Classification and Analysis of Multivariate Observations," Proc. 5th Berkeley Symp. on Math. Stat. and Prob. 1, Univ. of California Press, Berkeley and Los Angeles, pp. 281-297, 1967.

[39] Breiman, L., "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[40] "Awareness Survey on Life in Old Age for Before and After Retirement Generation," MUFG Financial Education Institute, 2019. (in Japanese) https://www.tr.mufg.jp/shisan-ken/pdf/kinnyuu_literacy_04.pdf, last accessed 2022/5/29.

[41] Statistics Bureau of Japan, Natl. Survey of Family Income and Expenditure, 2014. https://www.stat.go.jp/data/zensho/2014/index.html, last accessed 2022/5/29.

[42] Statistics Bureau of Japan, Consumer Price Index: https://www.stat.go.jp/data/cpi/index.html, last accessed 2022/5/29.

Notification, Wake-Up, and Feedback of Conversational Natural User Interface for the Deaf and Hard of Hearing

Takashi Kato*, Akihisa Shitara†, Nobuko Kato*, and Yuhki Shiraishi*
*Tsukuba University of Technology, Japan
Email: {a203101,nobuko,yuhkis}@a.tsukuba-tech.ac.jp
†University of Tsukuba, Japan
Email: theta-akihisa@digitalnature.slis.tsukuba.ac.jp

***Abstract*—Most voice-based conversational natural user interfaces (NUIs), such as Amazon Alexa and Google Assistant, rely on speech input and output, posing an accessibility barrier for the deaf and hard of hearing (DHH). For example, DHH users may not be aware of notifications from the system, may not receive response information, and the system may have difficulty recognizing their wake-words. In designing a conversational NUI for DHH users, we consider that simply replacing speech information with sign language information does not suffice to create an accessible, comfortable user experience. In this study, we conducted an experiment with 12 DHH users to determine whether luminous notifications and text display methods showing sign language in place of the standard text output were effective, as well as whether gazing was effective as a wake-up method. The second experiment was conducted with 24 DHH users to identify better wake-up and feedback presentation methods. We propose conversational NUI guidelines for DHH users based on the results of these experiments. We examined accessibility options for DHH users at each step of the conversation with the voice user interface (VUI), and expect this work to serve as a basis for future conversational NUI design.**

Keywords*—Deaf; hard of hearing; hearing impaired; sign language; accessibility.

I. INTRODUCTION

In this study, to propose design guidelines for a conversational natural user interfaces (NUIs) for dead and hard of hearing (DHH) users, we examined accessibility methods at each step of a conversation with voice user interfaces (VUIs). This study builds on and extends our previous work [1], in which we first introduced a conversational NUI for DHH users.

User interfaces (UIs) are a necessary medium for passing information between computers and humans. Research on NUIs designed to enable natural and intuitive operation by humans has progressed in recent years. Applications include touch panels, gesture control systems, and VUIs. For greater convenience and ease of use, "conversational" interaction with users is required to enable intuitive operation and mental support [2]. Advances in speech recognition, speech synthesis, and natural language processing have enabled humans to interact naturally with UIs [3] However, it remains difficult for DHH users to use and converse with VUI systems due to their inability to receive speech information as audio and high word error rate (WER). WER is a criterion used to evaluate speech recognition technology, which indicates the probability of words that could not be heard over the total number of words uttered by a human. The WER of a speech recognition system developed by Google averaged less than five percent in 2017 [4]. However, Bigham and others reported a WER for DHH users of 40% [5]. Abraham showed that current speech recognition technology is not yet usable by DHH users [6]. Moreover, he also reported that low WER could be overcome if more speech data from DHH individuals could be collected. However, collecting large amounts of speech data from DHH individuals would require considerable time and expense. As a result of the widespread use of Amazon Alexa and Google Assistant, voice control is becoming a ubiquitous interface technology. As this trend continues, an increasing need to address accessibility issues for DHH users in this technology is evident.

Accessibility studies on conversational user interfaces (CUIs) for DHH users have reported that sign language is more suitable than gestures and text as an alternative input method to replace speech [7]. It has also been reported that the use of sign language is preferable to touchscreens as an input method [8], and DHH users are interested in interacting with a system in sign language [9]. As a result, researchers in human-computer interaction (HCI) have begun to consider user interfaces that can interact with sign language [10], and the design and construction of sign language interfaces have been recognized as a notable research topic [11]. In recent years, researchers have begun to evaluate technologies for sign language recognition, generation, and translation from an interdisciplinary perspective [12], and this topic was addressed at a workshop on UIs [11]. Various calls to action were presented, including "develop user interface guidelines for sign language systems". However, basic research on how sign language users interact with computational systems remains relatively rare in the relevant literature. Although many systems have been developed for sign language users, most studies have focused on evaluating the systems themselves and did not outline the interaction principles between the user and system. As a result, each team developing a new system must design the interface almost from scratch, without the benefit of general design guidelines based on research. Therefore, design guidelines for NUIs should be developed, especially sign language-based NUIs, which have been the subject of considerable research.

In this study, we investigate whether graphical user interface (GUI) and VUI design guidelines can be used as a reference when developing guidelines for conversational NUI from the perspective of DHH users. The main GUI design guidelines

include “10 Usability Heuristics for User Interface Design” by Nielsen [13] [14], “Seven user-centered design principles” by Norman [15], and “Eight Golden Rules of Interface Design” by Shneiderman [16]. GUIs have long used heuristics as a key to implementing successful designs and avoiding usability problems. GUI design guidelines cannot be directly applied to VUIs [17]. VUIs and NUIs, including sign language conversations, use touchless interaction and natural language to interact with users. Therefore, it is challenging to devise NUI guidelines based on GUI design guidelines, including for sign language. Next, Alexa [18] and Google Assistant [19] are mainly considered in terms of VUI design guidelines. It has been reported as essential to establish a foundation of VUI principles to guide future designers in developing spoken conversation systems [20]. In addition, guidelines for VUIs are designed to assume speech-based interactions. The author believes that simply replacing speech information, mainly used when conversing with VUIs, with sign language information is not sufficient to create a comfortable user experience for DHH individuals. Therefore, to create conversational NUI guidelines from the perspective of DHH users, we consider alternative access methods for DHH users at each step of a conversation with a VUI. Based on the above, we propose guidelines for conversational NUI that are most suitable for DHH users based on the existing conversation steps with VUI.

From the conversational process between a listener and a conversational NUI (referred to herein as “the system”), we clarify the elements necessary to create a comfortable user experience for DHH users. Figure 1 shows the conversion process. At the beginning of the conversation, the user must invoke the system with a wake-word. A wake-word is a simple spoken phrase that triggers the AI assistant to accept speech input, such as “Alexa.” for Alexa, “OK, Google.” for Google Assistant, and “Hey, Siri” for Siri. The system then detects the user’s wake-up command and provides feedback that the system is ready to accept commands. For example, Alexa displays a blue bar at the bottom of the screen, and Google Assistant displays a bar at the top to provide feedback to the user’s command. The user then commands the system to access weather, news, alarms, etc. The system then executes the task according to the user’s commands. Other conversational situations include “alarm notifications” and “video calls” in situations where the system calls the user. We believe that simply substituting sign language for voice input is not sufficient to create a comfortable user experience for DHH users who mainly use visual information. Therefore, in addition to sign language, more optimal means of notification, information transmission, wake-up, and feedback presentation for DHH users must be identified.

First, we consider the notification method. Hearing persons can obtain audio information from the system without constantly looking at the system. For DHH users who cannot acquire audio information, it may be challenging to catch a response from the system when they are not looking at the display, even if they use a device with a display (Echo Show or Nest Hub). That is, the best output method of the system

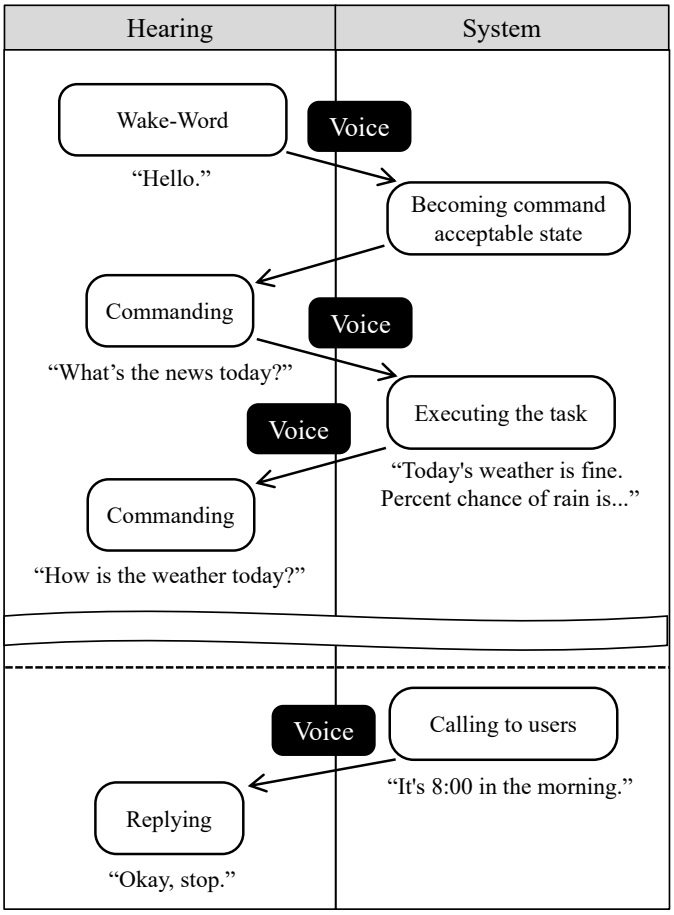


Figure 1. Activity of the conversation process between the Hearing and the system.

for DHH users should be investigated to identify an alternative to voice output provided by CUIs. By contrast, luminous notifications are familiar in Deaf culture. For example, DHH users use intercoms, alarms, and fire alarms with luminous notification functions in their daily lives [21]. In addition, a luminous device that transmits the direction of the sound source of the surrounding alarms to DHH users with light has been developed [22]. Figure 2 shows that we investigated whether luminous notifications could improve usability for DHH users.

Next, we consider the method of information transmission. Subtitling has recently become an accessibility feature for CUIs with displays [23]. Even if DHH users use this feature, there is a concern that the user experience may be worse if the system outputs subtitles, because an interaction will mainly be in sign language if sign language input from the user is enabled. With devices such as Alexa entering common usage in homes, there have been reports of increased hands-free interaction with devices placed in the kitchen or living room [24]. Therefore, it is conceivable that people may be more likely to interact with computational agents while doing other things. Therefore, DHH users should also be able to capture responses from CUIs while doing other things.

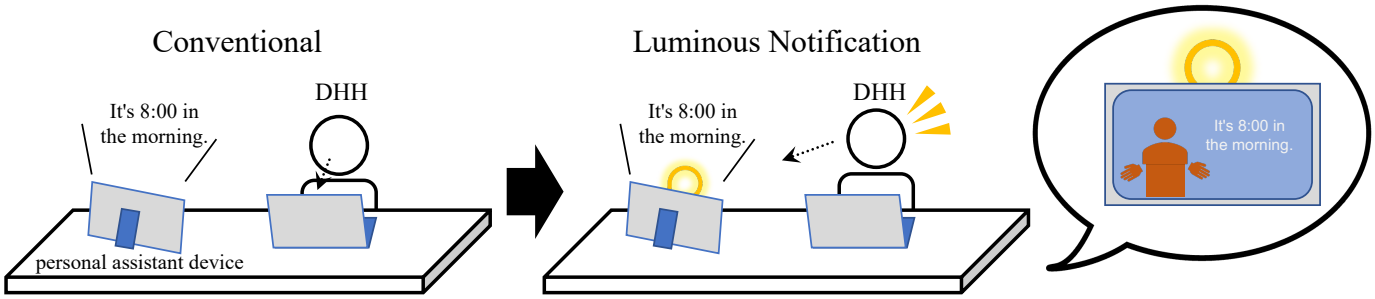


Figure 2. Hypothesis in light notification.

Here, clarifying whether the system's sign language or text output method affects the user experience of DHH users based on the difference between sign language flowing and text shown for a given time is essential. A study was conducted on the impact of sign language interpretation, subtitling, and the two together on the support of deaf students in secondary and higher education settings [25]. The study reported that the performance of the DHH students was significantly higher when only subtitles were used compared to the other two conditions. One study evaluated subtitles and sign language's combined effect in understanding television content [26]. Here, it was reported that providing subtitles and sign language was very helpful in improving the accessibility of TV content to more DHH individuals than providing each of the two alone. In addition, a study investigated whether incorporating sign language video into text-based web pages improved accessibility for DHH users [27]. The authors reported that information presented through sign language videos increased the interest of DHH users on the Web. However, all these studies were conducted for one-way media.

Conversational NUIs with two-way interaction require input from the user, and clarifying the preferences of users for sign language or text display methods is essential in such cases. Moreover, whereas designers must translate speech into the speaker's language in the case of television and the Web, CUIs are transmitted by a computer, so the language can be adjusted to suit the recipient. In other words, the designer should consider output methods using sign language or text along with users preferences in terms of attributes. Therefore, we investigate the necessity of sign language and subtitles for DHH use3rs under the condition of parallel work when CUIs provide both.

Then, we consider the wake-up method. DHH individuals need to make eye contact with a person they intend to talk to in order to start a conversation with them, and they do this by tapping their shoulder or waving [28] [29]. The preferred wake-up techniques of DHH users in descending order of preference include the use of the ASL sign name of the device, waving in the direction of the device, clapping, using a remote control, using a phone app, and fingerspelling the English name of the device [30]. Here, user preference for the sign name exceeded that of waving owing to the concern that the system would recognize unintended waving as a wake-up command. However, the use of eye contact, which is essential in starting a conversation in interpersonal communication with DHH individuals, has not been examined. "Eye contact" evokes and facilitates others' behavior and is involved in the initiation and progression of conversational interaction. We believe that gaze allows for a natural interaction without an explicit wake-up and increases user satisfaction. Based on the above, the possibility of using gazing in a system for DHH users should be considered, and the optimal wake-up method should be selected from among signing a name, waving, and eye gazing.

Finally, we consider how feedback is presented. In contrast to those with hearing, interpersonal communication with a DHH individual requires their attention when calling out to them. This suggests that hearing people need not confirm the feedback presented by the system compared to DHH individuals. It is becoming more common for listeners to enter commands in succession after a wake-word when operating such systems by voice. Hearing people assume that they are constantly being listened to by others, whereas DHH individuals typically believe they are only being listened to when others are looking. As for how Alexa devices present feedback, Echo Show 5 shows a blue bar, and Echo Show 10 shows a shaking head motion. It remains unclear whether these feedback methods will improve the user experience for DHH users. Therefore, investigate feedback presentation for DHH users should be investigated to consider better feedback methods.

To create a comfortable user experience for DHH users, this study examined the following research questions, and we propose guidelines based on the results.

- RQ1: Does the light-based response of the CUI improve usability for DHH users?
- RQ2: What is the best sign language/text display method for CUI for DHH users?
- RQ3: Is gaze tracking an effective method of waking up a CUI for DHH users?
- RQ4: Would DHH users prefer to see an indication from the system that it is ready to accept commands before giving them?
- RQ5: What might be a better way for DHH users to wake up a system?
- RQ6: Is there a better way for the system to indicate that

it is ready to accept a command when it detects DHH users' wake-up commands?

We surveyed students at the Tsukuba University of Technology [33] to investigate these research questions. The Tsukuba University of Technology is a university for the visually and hearing impaired. The hearing level of the DHH students is generally 60dB or higher in both ears. Tsukuba University of Technology students include not only deaf but also hard of hearing students.

This study contributes empirical knowledge regarding the preferences and concerns of DHH users regarding features such as notification, information transmission, wake-up, and feedback presentation, aiming to provide useful guidance for future system designers.

The remainder of this study is organized as follows. In Section II, we describe related studies on smart speakers and gesture interfaces and the Wizard of Oz method. In Section III, we investigate whether a sign language conversation system using luminous notification and gaze tracking can improve usability for DHH users to address RQ1–RQ3. In Section IV, we examine wake-up and feedback methods for DHH users to address RQ4–RQ6. In Section V, we propose design guidelines for conversational NUIs for DHH users based on our investigation of these six research questions. In Section VI, we describe the limitations of this work and suggest some avenues for future research. Finally, in Section VII, we provide some concluding remarks.

II. RELATED WORK

To demonstrate the effectiveness of sign language-based conversational NUIs, we focus on two other NUIs that allow human-computer conversations: smart speakers and gesture interfaces. Then, we describe the Wizard of Oz method used to construct our experimental environment.

A. Diffusion Status and Challenges of Smart Speaker

In the latter half of the 2010s, with the improvement of voice recognition performance and AI technology, various companies released smart speakers to be used in the home [34]. Conversational AI agents that play music, news, and weather forecasts in response to natural language questions from the user have become widely popular. Using these devices, a user can perform a task by speaking a wake word into a nearby microphone or speaker unit and then continuing with a question or request. The reasons for the widespread use of smart speakers include improvements in voice recognition and natural conversation technology as well as their hands-free operation, which allows users to avoid interrupting other activities [35]. According to a Canalys survey, worldwide shipments of smart speakers were expected to reach 320 million units in 2020 and 640 million units by 2024 [36].

However, smart speakers are limited in that they cannot be used in offices or public spaces and are difficult to understand in noisy environments. The former is the case because hearing is a passive and unconscious stimulus compared to vision [37]. Therefore, as long as others are nearby, the sound of a voice may be heard in any direction. Therefore, based on sign language conversation, NUIs are expected to solve the privacy problem in offices and public spaces. The latter does not require speech recognition, and thus does not cause recognition problems in noisy environments.

B. Application Examples and Challenges of Gestural Interfaces

Gestural interfaces use visual and bodily functions such as arms, fingers, and facial expressions. Examples of applications include Motion Sense on the Google Pixel 4 and several such methods to control drones. Motion Sense allows users to operate their phones by holding their hands over them without directly touching them, such as moving forward or backward in list of tracks playing music or stopping incoming calls or alarms. It is also equipped with a motion-sensing function that lights up the screen when the user places their hand near the phone. In combination with face recognition, users can quickly unlock the phone.

In this context, there are three challenges with gestural interfaces. The first is the "cognitive load" [39] [40] [41]. As gesture input is not linguistic, it is necessary to memorize commands, as in a command-line interface (CLI). Although there is a possibility that the object or situation can be suggested to some extent compared by the CLI, the more it relies on gesture operations, the more gestures the user must remember. Therefore, the gestures used as commands tend to be inconsistent. The second is "distinction of intention" [42]. One of the selling points of gesture manipulation is the intuitive and natural behavior that humans often perform. However, this intuition and naturalness overlap with our daily behavior. The problem arises when a gesture is mistakenly recognized as a command and executed without distinguishing between the actual operation and the everyday behavior. The third problem is "physical fatigue" [43]. The human body may be burdened by the movements used to perform gestures, such as moving the arm up and down, left and right, or raising the arm for a long time.

Therefore, NUIs based on sign language conversations are expected to overcome the challenges of gesture UIs, such as reduced expressive power and increased memory load, because they use natural languages for interaction. It is also suggested that for native signers, conversing in sign language is independent of the physical fatigue issues inherent in gestural interfaces.

C. Wizard of Oz

The recognition rate of a real-life continuous sign language recognition system developed in 2019 was 39.6% [44]. Therefore, it was impossible to conduct experiments incorporating sign language recognition technology to interact with a user interface using sign language. We used the Wizard of Oz method is a solution to this problem [45] [46]. With this method, a human referred to as a wizard pretends to act as a computational system and interacts with the user. In the Wizard of Oz method, even if the entire system is not yet

TABLE I
LINGUISTIC MODALITIES OF HEARING AND DEAF INTERACTIONS WITH CUIS. "*" :RQ1,"**" :RQ2,"***" :RQ3

User \ Type	Conversation		Call	
	System→User	User → System	System→ User	User →System
Hearing	Voice		Voice	
DHH	Sign Language/Text **		Luminous *	Eye Gaze ***

"*" :RQ1, "**" :RQ2, "***" :RQ3.

complete, the wizard can complement the undeveloped parts of the system and allow it to seem to perform these functions for the users. In this study, we use this approach to experiment with the behavior of a sign language recognition system to extract data generated in situations where people are speaking with sign language.

III. SIGN LANGUAGE CONVERSATIONAL USER INTERFACES USING LUMINOUS NOTIFICATION AND GAZE DIRECTION

A. Research Questions

In Section III, we consider the research questions RQ1-3. As shown in Table I, RQ1–RQ3 address the mutual input and output modalities that DHH users may prefer to achieve when interacting with CUIs. DHH users use sign language/text modalities when interacting with CUIs (RQ2). They use the luminous notification modality when calling from CUIs to DHH users (RQ1). In contrast, when DHH users call to (wake-up) a CUI (RQ3), they use the gaze modality. We investigate whether using these mutual input/output modalities in CUIs can improve the user experience of DHH users. To answer RQ1 to RQ3, we constructed a sign language conversation system with optical notifications based on the Wizard of Oz method and conducted an experimental evaluation (Experiment 1).

B. Methodology

1) Participants: Using a mailing list, we solicited the cooperation of DHH students in their 20s to participate in the Experiment 1. 12 students ultimate participated.

We also investigated the characteristics of the participants to analyze the effect of their attributes on the results of the experiment. Specifically, we conducted a preliminary questionnaire survey on the age, gender, and cochlear implant/hearing aid use of the participants to determine whether they use their voices when communicating, whether they mainly use auditory or visual communication, and whether they use both, as well as their experience with learning sign language and their experience using VUIs. Figure 3 shows the results. The age of our participants, 8 males and 4 females, ranged between 20 and 24. We asked the participants to rate their experiences with using VUIs on a 4-point Likert scale (1 = usually, 2 = sometimes, 3 = rarely, and 4 = never). The results showed that the response of one participant was 2, four participants responded with 3, and seven participants

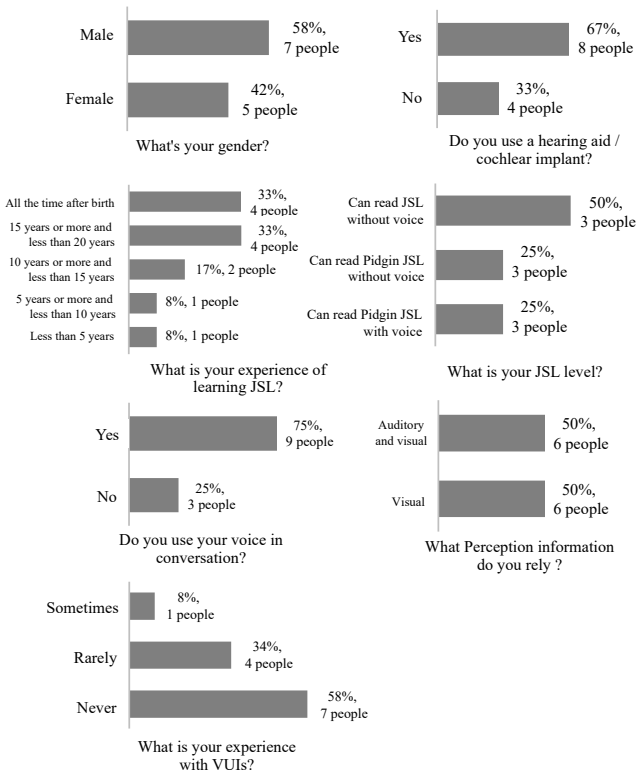


Figure 3. Characteristics of Participants (N=24) in Experiment 1.

responded with the value 4. The majority of the participants commented that "their voices could not be recognized" or that "they lived without speaking verbally." Research mentioning that a minimal number of DHH users use personal-assistant devices [47] indicates a similar trend to that exhibited by the participants in this experiment.

Experiment 2 was approved by the Research Ethics Review of the Tsukuba University of Technology, where the experiment was conducted. The duration of the Experiment 2 was 90 min, and the honorarium paid to the participants was 1,305 yen (approximately $12).

2) System Architecture: The basic configuration of the system constructed in Experiment 1 comprised an iPad, a Meross Smart Wi-Fi LED Bulb (LED Bulb), and a GoPro HERO9 camera. Figure 4 shows the appearance and operation of the system. We set four tasks that the system can perform: "Phone call," "Alarm settings and notifications," "Checking the Weather," and "Checking the News." An Apple iPad simulated Alexa, and the display was created using Microsoft PowerPoint 2019 and combined with the signer's video. To switch screens remotely, the remote function of Keynote was used. We included LED bulbs that could be set to any color (16 million RGB colors) and a blink cycle, and controlled the system remotely from a smartphone app. An LED bulb flashes yellow when the system notifies the user of a "Phone call" or "Alarm notification" and emits a light green when the system provides "Weather" or "News" to the user. In addition, the

Figure 4. System Prototype in Experiment 1.

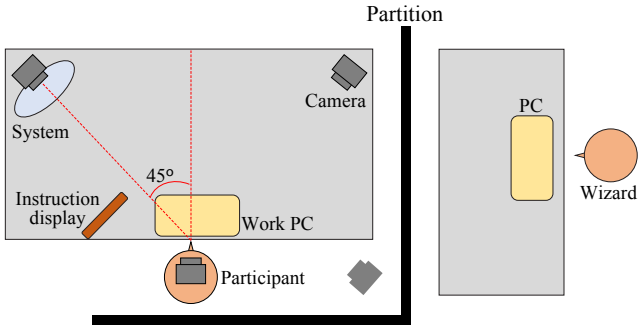


Figure 5. Experiment 1 Setup.

GoPro camera was used to view the sign language input from the user.

3) Procedure: Figure 5 shows the Experiment 1 environment. In the environment of Experiment 1, we assumed that the participants interacted with the system while working on their PCs. Therefore, we placed the system on the left side of the desk in front of the participants at 45 degrees, with a workstation PC in front of them. We tried to make the participants aware of the system response while the system was operating so that they did not have to constantly look at the system. We also aligned the system at the eye level of the participants. The instruction device prompted the participant to issue commands to the system at certain times. We incorporated a program in PsychoPy (v2021.1) [48] to display numbers and/or English letters at random positions on the screen. In addition, the frame rate of the installed camera was 50 FPS.

The critical points for the participants and the experimenter (Wizard) in this environment were as follows.

Participant

1) Owing to the nature of the Wizard of Oz method, the participants assumed that aa computational sign language recognition system was being used, and did not know that a person (wizard) was operating the system.
2) During the experiment, the participant were asked to continuously work on the task of "entering numbers and English letters displayed at random positions on the screen of a workstation PC with the keyboard as they appear."
3) The participants commanded the system using sign language commands for "Setting the alarm," "Checking the weather," and "Checking the news." The participant pressed the button as soon as they noted the end of the description of "Weather" or "News" from the system.
4) During the work, the participants used sign language commands to stop the "Alarm notification" and "Phone call" sent by the system.
5) During the experiment, the users wore a GoPro attached to a head strap mount.

Experimenter (Wizard)

1) Owing to the nature of the Wizard of Oz method, the experimenter was required to avoid letting the participants know that the experimenter is operating the system when performing the sign language recognition system.
2) During the experiment, the experimenter operated the system and the LED bulb.
3) When we asked the participants to conduct a specific task at an arbitrary time, we showed them the content of the task and an example of the command to be performed, and we immediately turned off the screen after confirming that the participants understood the task.

Before the experiment, we explained how to use the system and how the system was designed to behave for each of the four tasks. In addition, to familiarize the participants with command execution using sign language, we asked them participate in a practice session to perform a task equivalent to the real one before the actual experiment was conducted. The participants conducted each of the four tasks once and repeated them twice. To eliminate order effects, the order of the tasks for each participant and the two conditions, "Luminous/Conventional," were counterbalanced.

4) Analysis Method: For a time analysis using video, we applied the ELAN [49] tool.

For RQ1, we used the system usability scale (SUS) [50], a widely applied evaluation index for a quantitative evaluation of usability, to examine the usability of "Luminous" and "Conventional." In addition, we believe that improved usability is also related to awareness. To evaluate the awareness of the notifications from the system, we measured by video the time between the notifications were provided and when the participant noticed and reacted to them. We defined the reaction time for a "Call" as the time between the change in the display screen as the reaction starting point and the users turning their eyes to the screen as the reaction endpoint. However, if the light turned on before the screen changed, the reaction starting point was when the light turned on. We defined the reaction time for a "Response" as the time between the change in the display screen and the user pressing the button. However, if the light turned off before the screen changed, the reaction starting point was defined as the time when the light turned off.

For RQ2, we examined the participants' need for sign language/text. After the experiment, we administered a questionnaire to determine the need for sign language/text using

a five-point Likert scale (1 = agree, 2 = agree a little, 3 = neutral, 4 = disagree a little, and 5 = disagree).

For RQ3, we examined whether the participants gazed at the system before giving a command in sign language. For this purpose, we measured the percentage of the total number of times the participants gazed at the system at least once in the 5 s before the sign language command and the time between the start of the gazing and sign language using video. For the data to be analyzed, there was a scene during the experiment in which the system responded to an "Alarm notification" or "Phone call," and the user issued a command to stop the system. The users looked at the response screen before making a sign language command, we did not collect analysis data to investigate whether they gazed at the screen before the sign language command. The three tasks for which the user actively gave a sign language command were used as data for analysis, i.e., "Setting the alarm," "Checking the weather," and "Checking the news."

C. Results

1) System Usability Scales: Figure 6 shows the results of the SUS investigated after the experiment. The mean SUS value of "Luminous" was 80.67 (SD 7.62), and that of "Conventional" was 68.96 (SD 14.6). As a result of the Wilcoxon signed-rank test, "Luminous" was found to be significantly higher ($p < .05$).

2) Reaction Time: Figure 7 shows the results of the reaction time. The mean reaction time to "Alarm notification" and "Phone call" of "Luminous" was 0.91 s (SD 0.35), and the mean value of "Conventional" was 1.19 s (SD 0.57). The Wilcoxon signed-rank test showed that the reaction time was significantly shorter for "Luminous" ($p < 0.01$). The mean reaction time to the end of "Weather" and "News" for "Luminous" was 1.37 s (SD 0.50) s, and the mean value of "Conventional" was 1.91 s (SD 1.22). The Wilcoxon signed-rank test showed no significant differences between "Luminous" and "Conventional" ($p > 0.05$).

3) Necessity of Sign Language/Text: Figure 8 shows the results of a 5-point Likert scale used to assess the need for sign language and text for the 12 participants, respectively. In terms of sign language, we found the following. 1: "Agree" was reported by four participants. 2: "Agree slightly" was reported by two participants. 3: "Neutral" was reported by three participants. 4: "Disagree slightly" was reported by two participants. 5: "Disagree" was reported by one participant. There were three participants who did not need sign language (4,5), and their sign language experience was, in order of shortest to longest, three years (1st), five years (2nd), and fifteen years (5th). By contrast, for text, "1 = Agree" was reported by nine of the participants, and "2 = Agree slightly" was reported by three of the participants.

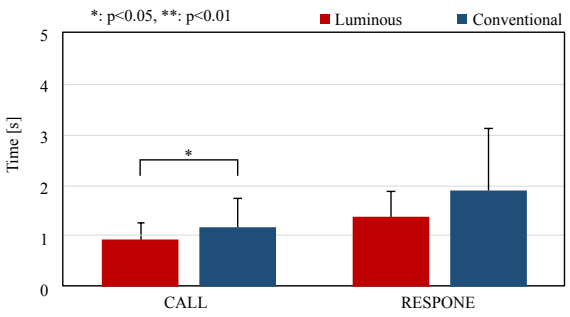


Figure 7. Reaction time for each feedback from the system.

4) Gaze Direction: During the experiment, a pattern occurred in which the experimenter turned off the screen of the instructional device late, indicating the task to be performed, and the participant gave a sign language command while reading. We removed these data from our analysis because they were unsuitable for examining whether the participants were gazing at the system. The participants (N = 12) input sign language commands into the system 69 times: 23 times for "Setting the alarm," 24 times for "Checking the weather," and 22 times for "Checking the news." Table II lists the percentage of the total number of times the participants gazed at the system at least once during the 5 s before the sign language command and the average time from the start of gazing to the start of the sign language, as well as the standard deviation and minimum and maximum values. A high percentage of the total number of users gazed at the system before using the sign language commands.

Three participants, P3, P8, and P9, waved before using sign language. These three participants had experience using

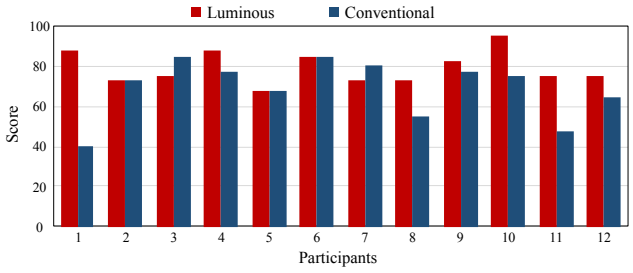


Figure 6. SUS score for each participant (N = 12).

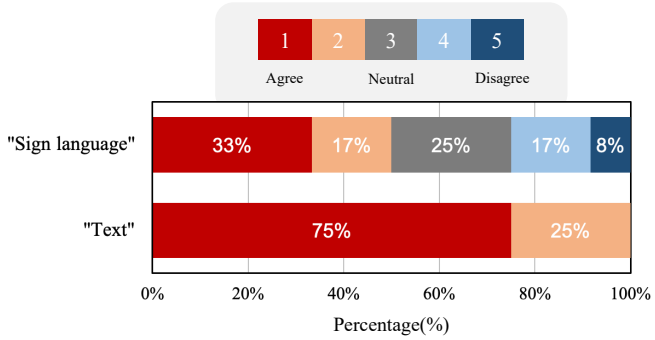


Figure 8. Necessity of "Sign language" / "Text".

TABLE II
PERCENTAGE OF EYE GAZE, MEAN AND STANDARD DEVIATION OF TIME OF EYE GAZE

Task	*Percentage* (%)	*Mean*±*SD* (*s*)	*Min* (*s*)	*Max* (*s*)
Alarm	100	0.76±0.61	0.20	3.18
Weather	100	0.43±0.23	0.10	1.08
News	86.4	0.59±0.44	0.20	2.08
Total	93.4	0.59±0.47	0.10	3.18

VUIs and knew that they should use a waking command. The interviews also revealed that they thought it was necessary to take explicit action before talking to the system during this experiment.

D. Discussion

1) RQ1: Efficacy of Luminous Notification: The results described in Section III-C1 suggest that luminous notification improved usability for DHH users in noticing notifications from the system. In addition, the reaction times to "Alarm notification" and "Phone call" were significantly shorter when using a luminous notification, suggesting that it is easier to notice such notifications from the system.

Participants commented, "I am familiar with luminous notification methods, such as the intercom system in my house, which notifies me by light, so it would be more impressive to add light to the system as well. I can notice the light notification even when I am concentrating on my work." However, there were also comments such as "I feel uncomfortable with the luminous notification because I live my life relying on sound. Therefore, the system may not be suitable for people who use their daily hearing functions.

From Figure 6, we can see that the usability of P3 and P7 decreased with a luminous notification. The participants commented that they did not feel the need to use a luminous notification because they only noticed the change in the system screen. This may have occurred because there were cases in which DHH users could respond to conventional methods [51]. In this experimental environment, the system was placed on the left side of the desk in front of the user at a 45-degree angle within their peripheral vision. During this experiment, we placed the system within the peripheral vision of the front of the user, and thus some of the subjects noticed changes in the screen without looking at the system.

In contrast, there were no significant differences in the reaction time to the end of the "Weather" and "News" responses when using the luminous notification, as described in Section III-C2. However, some of the participants commented positively that "it was convenient to know when the response ended without having to look at the system." By contrast, others commented negatively that "luminous notification was not necessary for the information (weather and news) that I wanted," and "the light was too bright." As a result, we found that the usability of the system could be improved by reducing the light exposure and improving the luminous notification method, although the noticeability remained the same.

A participant commented that it would be preferable to increase the brightness of the display, as in ON AIR, instead of directly informing the user with LED bulbs. For a luminous notification, we used LED bulbs, which were initially used as lighting fixtures. Therefore, a way to change or vary the brightness of the display directly should be considered instead of using external LEDs.

2) RQ2:How to Display Sign Language/Text Suitably: From Section III-C3, it may be observed that all of the participants needed to display text regardless of the user attributes. By contrast, the necessity for the sign language display varied from participant to participant. In addition, it may be observed that those who had not signed for a long time tended not to believe that sign language was necessary.

The participants who did not need sign language commented that they did not understand sign language and had trouble processing information when both sign language and text were output simultaneously. By contrast, the participants who needed to use sign language commented that the sign language display made it easier for them to remember the system responses. Some of the participants commented, "When I look at the task screen while working, the remaining text is better than the flow of the sign language."

For hearing users, interaction with VUIs has the advantage of being eyes-free [52]. Therefore, users frequently interact with conversational NUIs while performing other tasks. However, in the case of DHH users, the advantage of eyes-free interaction is lost because they cannot acquire audio information and instead gaze at the screen. To complement this, we anticipate that DHH users would need text information that they can recognize, even if they look away for a moment. One possible solution to this problem is to stop the sign language flow when the user looks away and resume when the user returns their gaze to the screen.

3) RQ3:Efficacy of Gaze Direction: Section III-C4 shows that the participants tended to gaze at the screen before speaking in sign language.

During this experiment, we did not provide instructions on how to wake-up the device. Nevertheless, the participants naturally gazed at the system with a high probability.

By contrast, 3 of the 12 participants did not gaze at the system but waved instead. When DHH users use waving as a wake-up method, there is a concern that signs made while talking to another person may be recognized as waving at unexpected times, such as during a "Phone call" or "Alarm notification." In addition, we believe that gazing is a more natural manner of interacting than waving each time a command is used. These results suggest that directing one's gaze to a device may be considered a compelling wake-up method.

When Alexa waits for a response from the user, there is a time limit of 8.0 s [53]. From Table II, the maximum time between gazing at the system and the start of sign language was 3.18 s. That is, when DHH users used gazing as a wake-up method, they could use commands within the system's waiting time.

IV. CONSIDERATION OF WAKE-UP METHOD AND FEEDBACK METHOD

A. Research Questions

In Section IV, we consider research questions RQ4–RQ6, which cover the input and output methods for DHH users at the beginning of a conversation with the conversational NUIs. First, we investigated the need for feedback on the system for DHH users (RQ4). Then, we investigated the best method to wake up the system from DHH users (RQ5). We investigated the best way to present the feedback to the user to indicate that the system has detected the user's wake-up command and is ready to accept commands (RQ6). To answer RQ4–RQ6, we constructed a conversational system that presented a variety of feedback based on the Wizard of Oz method and conducted an evaluation experiment (Experiment 2).

B. METHODOLOGY

1) Participants: Using a mailing list, we solicited the cooperation of DHH students in their 20s to participate in Experiment 2. 24 students participated.

We also investigated the characteristics of the participants to analyze the effect of their attributes on the results of the experiment. Specifically, we conducted a preliminary questionnaire survey on the age, gender, and cochlear implant/hearing aid use of the participants to determine whether they used their voices when communicating, whether they used both, and their experience of learning sign language, their identity, and their experience using VUIs. Figure 9 shows the results. The age of our participants, 14 males and 10 females, ranged between 20 and 23. We asked the participants to rate their experience of using VUIs on a 5-point Likert scale (1 = everyday, 2 = several times a week, 3 = several times a month, 4 = less than once a month, and 5 = never). As a result, the most significant number of 20 participants answered "5:Never," accounting for 83% of the total. Research mentioning that a minimal number of DHH users use personal-assistant devices [47] has indicated a similar trend to that of the participants in this experiment.

Experiment 2 was approved by the Research Ethics Review of the Tsukuba University of Technology, where the experiment was conducted. The duration of Experiment 2 was 90 min, and the honorarium paid to the participants was 1,500 yen (approximately $13).

2) Experimental conditions: **Wake-up**

We compared the gaze-based method validated in Section IV with other wake-up conditions and identify the preferences of DHH users.

I1: Eye Gaze

In the reports of studies on wake-up commands for DHH users, signing a name was rated as the highest condition. However, the authors did not consider gaze direction in their comparison. Therefore, in Experiment 2, we investigate whether adding gaze tracking to the wake-up method changed the preferences of DHH users.

I2: Sign-name

Assuming that the system installed was Alexa, and referring to the previous research, the method of signing a name involved using an "A" handshape used to draw an "X."

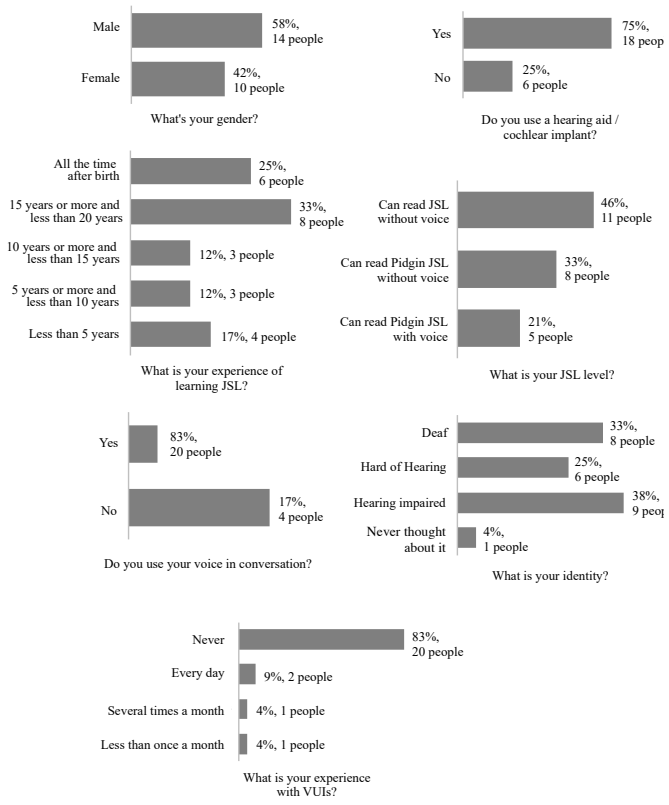


Figure 9. Characteristics of Participants (N=24) in Experiment 2.

I3: Waving

A wake-up wave is a left-right-forth motion in which the user holds their palm toward the system. According to reports on wake-up research [30], the evaluation of the wake-up was lowered due to the concern that the system would recognize unintentional waving as a wake-up gesture. Therefore, we thought we could gain new insights by comparing the results with the gaze-based system, in which the user's intention to wake up the system is clearer.

Feedback method

O1: Blue bar and low-intensity

The Echo Show recognizes a wake word and displays a blue bar at the bottom of the screen. The screen's brightness is lowered to make the blue bar stand out to show that the device is ready to process the user's request [54]. We added a baseline condition to explore whether these conventional response presentation methods are desirable for DHH users.

O2: Sign language added to O1

As DHH users mainly use sign language in their conversations, we believe it would be optimal for DHH users to have the system respond in sign language. For this reason, we added a sign language response.

O3: Shaking head motion added to O1

To investigate whether a head-shaking motion, like the Echo

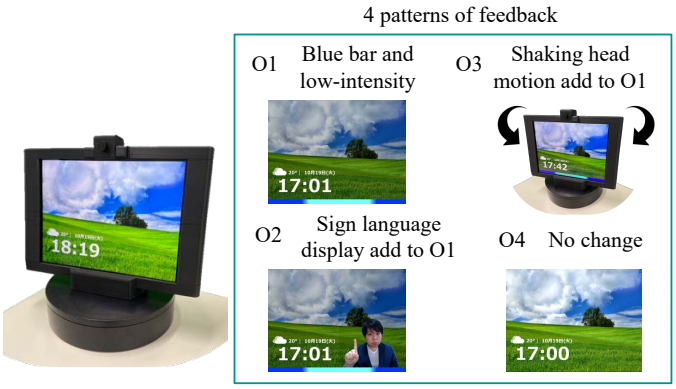


Figure 10. System Prototype in Experiment 2.

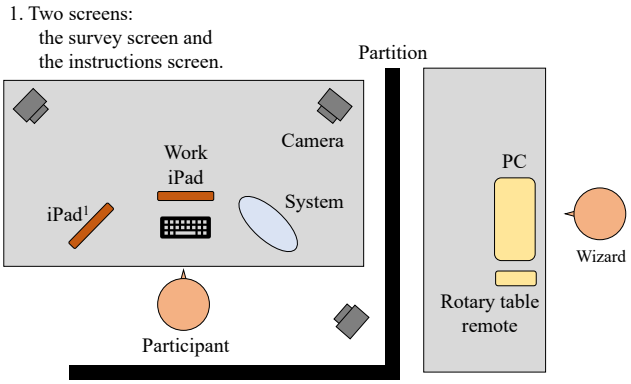


Figure 11. Experiment 2 Setup.

Show10 [55], could improve usability by DHH users, we added a response with a head-shaking motion.

O4: No change

To investigate the need for feedback for DHH users of RQ4, we added a condition of not presenting feedback, indicating no change to the system, in contrast to O1–O3.

3) System Architecture: The basic configuration of the system constructed in Experiment 2 comprised an iPad and an electric rotary table. Figure 10 shows the appearance and operation of the system. The screen size of the iPad was 10.2 inches, and that of the Echo Show10 was 10.1 inches, which was almost identical. For the system's design, Echo Show 10 was simulated, the display was created with Microsoft PowerPoint 2019, and the cover around the iPad display was created with a FlashForge Adventure 3 FFA-103 3D printer. In addition, the remote function of Keynote was used to switch the screen remotely. For the sign language display, we synthesized a sign language movie called "What's up?". To add the function of the head-shaking motion, we used an electric rotary table with a diameter of 20 cm and a load capacity of 20 kg manufactured by BAOSHISHAN. A remote controller was used to remotely control the rotation speed and direction angle of the table.

4) Procedure: Figure 11 shows the Experiment 2 environment. In the Experiment 2 environment, we assumed that the participants interacted with the system while working. Therefore, we placed the system on the right side of the desk in front of the participants at a 45-degree angle, the work iPad in front of them, and the iPad with the questionnaire and instruction screens split on the left side of the desk in front of the participants at a 45-degree angle. Question on user satisfaction for both the wake-up and feedback conditions, the ranking of satisfaction for each, and the need for feedback were created in Forms using a 7-point Likert scale. The instruction screen was created in Keynote to show the wake-up procedure to be performed by the participant and the feedback to be provided by the system. In addition, the frame rate of the installed camera was 50 fps.

The critical points for the participants and the experimenter (Wizard) in this environment are summarized as follows.

Participant

1) Owing to the nature of the Wizard of Oz method, the participants assumed that a sign language recognition system was in use, and did not know that a person (wizard) was operating the system.
2) During the experiment, the participants were asked to continuously work on the task of "entering numbers and English letters displayed at random positions on the screen of a workstation PC with the keyboard as they appear."
3) The participants confirmed the wake-up procedure and the feedback provided by the system on the instruction screen.
4) After the participant typed about three words, they woke up the system, and as soon as the system feedback was presented, they selected one of the commands, "Check weather" or "Check news", and commanded the system in sign language.
5) After confirming that the system had finished answering the "weather" and "news" questions, the user completed the "satisfaction with both the wake-up method and the system feedback" questionnaire.

Experimenter (Wizard)

1) Owing to the nature of the Wizard of Oz method, the experimenter was required to avoid letting the participants know that the experimenter was operating the system when performing the sign language recognition system.
2) During the experiment, the experimenter controlled the system.
3) When we asked the participants to conduct a specific task at an arbitrary time, we showed them the content of the task and an example of the command to be performed, and we immediately turned off the screen after confirming that the participants understood the task.

Before the experiment, we explained to the participants how to use the system and how the system behaved. In addition, to familiarize the participants with wake-up and command execution using sign language, we provided a practice session in which they performed a task equivalent to the real one before

the actual experiment was conducted. Participants performed each of the 12 conditions once, for 12 repetitions, combining the three wake-up conditions and the four feedback conditions. The order of the 12 conditions was determined using a Latin square design to eliminate order effects.

5) Analysis Method: To investigate the optimal combination conditions from the three conditions of the wake-up method (I1–I3) and the four conditions of the feedback method (O1–O4), we used a questionnaire with a Likert scale of 7 levels of satisfaction (3. very satisfied, 2. satisfied, 1. slightly satisfied, 0. neither satisfied nor dissatisfied, -1. slightly dissatisfied, -2. dissatisfied, -3. very dissatisfied). We asked the participants to respond to a questionnaire based on the Likert scale. From the response data (N=24), better wake-up conditions and feedback conditions were clarified. After completing the 12 conditions, the participants were asked to rank their satisfaction with the wake-up method (I1–I3) and the feedback method (O1–O4). The reasons for their satisfaction were investigated in an interview. To evaluate the effectiveness of the system in conversations with the participants, we conducted a video analysis of their behavior using the ELAN [49] tool. The minimum video measurement time was 0.02 s.

1) Time between the beginning and end of the wake-up action
2) Duration of gaze during waving
3) Time from the end of the wake-up operation to the beginning of command input

Then, to investigate the necessity for the presentation of feedback in RQ4, we asked the participants to respond to a questionnaire with seven levels of necessity (1. strongly agree, 2. agree, 3. agree a little, 4. neutral, 5. disagree a little, 6. disagree, 7. strongly disagree). The questionnaire was administered using a 7-point Likert scale. To investigate the effectiveness of gazing in the wake-up method of RQ3, we asked the participants whether they would like to perform I1 (eye gaze) together with I2 (sign name) and I3 (waving) in the wake-up method. We categorized the participants' responses into three patterns (1. like, 2. limited like, 3. dislike).

C. Results

1) Effect of the feedback: Figure 12 shows a summary of the mean and standard deviation of satisfaction of each feedback method (O1–O4). The mean satisfaction values were 1.90 (SD 1.16) in O1, 1.67 (SD 1.21) in O2, 1.67 (SD 1.55) in O3, and -0.18 (SD 1.64) in O4. Multiple comparisons using the Tukey's test were conducted to determine whether there was a significant difference in satisfaction between the four levels (O1–O4). The results showed that the level of satisfaction in O4 (no change) was significantly lower than that in the other feedback presentations (O1–O3, change) ($p < 0.01$).

Figure 13 shows a summary of the mean and standard deviation of the time from the end of wake-up to the beginning of command input for each feedback presentation condition (O1–O4). The times of the mean and standard deviation were 2.60 s (SD 1.06) for O1, 2.92 s (SD 1.31) for O2, 2.57 s (SD 1.17) for O3, and 1.79 s (SD 1.14) for O4. We conducted multiple comparisons using Tukey's test to check for a significant difference in time between the four levels (O1–O4). The results showed that the time in O4 (no change) was significantly shorter than the time in the other feedback presentation (O1–O3, with change) ($p < 0.01$).

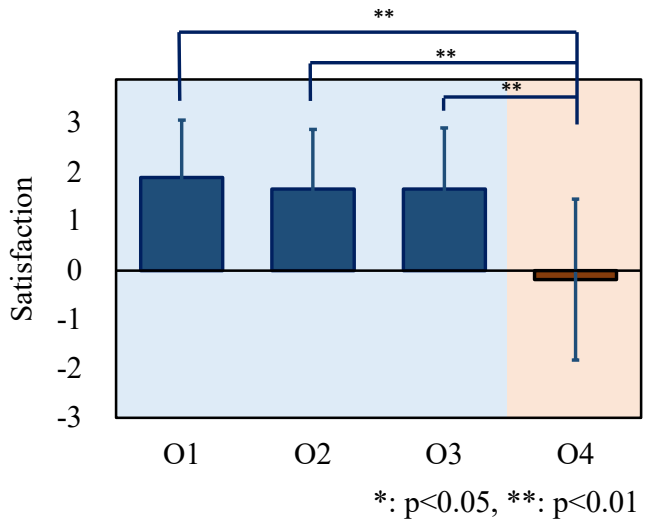


Figure 12. Mean and standard deviation of individual satisfaction with feedback.

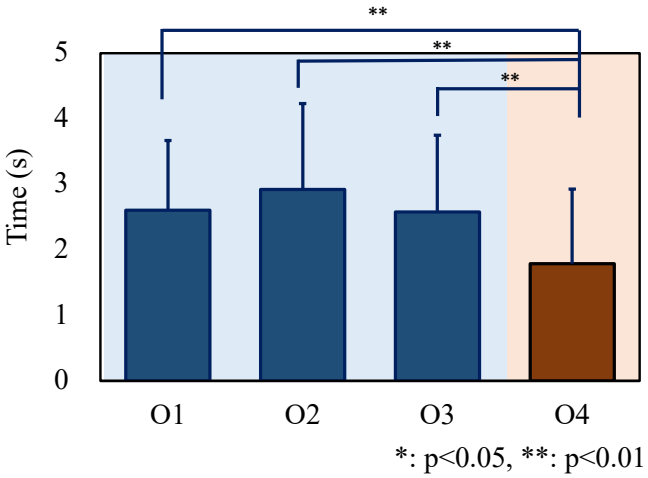


Figure 13. Mean and standard deviation of time from the end of wake-up to the beginning of command input for each feedback.

2) Satisfaction: Figure 14 shows a summary of the mean and standard deviation of the satisfaction for each of the nine conditions combining the wake-up and the feedback methods. The highest mean satisfaction of the input and output methods was 2.13 (SD 1.08) for the combination of I3 (waving) and O1 (blue bar and low-intensity). The lowest was 1.21 (SD 1.41)

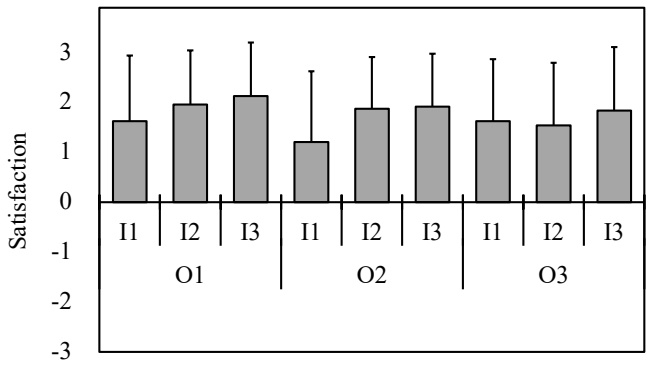


Figure 14. Satisfaction for each 12 conditions(N=24).

for the combination of I1 (eye gaze) and O2 (sign language add to O1) (SD 1.41).

We conducted a two-way analysis of variance (ANOVA) using the wake-up method (I1–I3) and the feedback method (O1–O3) as factors. The results showed that the main effect for satisfaction with the wake-up method was significant ($p < 0.05$), but the main effect for satisfaction with the feedback method was not significant ($p > 0.05$). In addition, the interaction between the satisfaction of the wake-up method and the feedback method was not significant ($p > 0.05$). For the wake-up method for which the main effect of satisfaction was significant, the mean satisfaction values were 1.49 (SD 1.32) in I1 (gazing), 1.79 (SD 1.13) in I2 (signing a name), and 1.96 (SD 1.13) in I3 (waving). Multiple comparisons of the Tukey's test were conducted to examine the difference in the mean satisfaction values among the three levels of the wake-up method. The results showed that the satisfaction level was significantly higher for I3 (waving) than I1 (gazing) ($p < 0.05$).

We conducted a two-way ANOVA of satisfaction using the wake-up method (I1–I3) and the feedback method (O1–O3) as factors for each characteristic. Table III shows a summary of the significance probabilities of the wake-up method, feedback method, and interaction for each characteristic. Four attributes were analyzed: "Whether the user was using hearing aids or cochlear implants," "Japanese sign language (JSL) level," "Whether the user was using voice in the conversation," and "Identity." The interaction was not significant for all characteristics ($p > 0.05$). The main effect characteristic for "other using aids/cochlear implants" was the wake-up method for participants who answered "no." These participants had the characteristic of "not relying on auditory information." The mean satisfaction values were 0.78 (SD 1.40) in I1 (gaze), 1.83 (SD 0.99) in I2 (sign name), and 2.06 (SD 1.00) in I3 (waving). We conducted multiple comparisons of the Tukey's test to examine the difference in the mean satisfaction values among the three levels of the wake-up method. As a result, we found that the satisfaction level of I2 (sign name) was significantly higher than that of I1 (gazing) ($p < 0.05$), and the satisfaction level of I3 (waving) was significantly higher than that of I1 (gaze) ($p < 0.01$). For the "JSL level," the characteristic that showed the main effect was the wake-up method for participants who answered, "I can read JSL without voice." These participants had a "high sign language level" characteristic. The mean satisfaction scores were 1.18 (SD 1.38) in I1 (gazing), 1.94 (SD 0.97) in I2 (signing a name), and 2.21 (SD 1.02) in I3 (waving). Multiple comparisons of the Tukey's test were conducted to examine the difference in the mean satisfaction values among the three levels of the wake-up method. As a result, we found that the satisfaction level was significantly higher for I2 (sign name) than for I1 (gaze) ($p < 0.05$), and the satisfaction level was significantly higher for I3 (waving) than for I1 (gaze) ($p < 0.01$). Concerning "Whether the user was using voice in conversation," the characteristic that showed the main effect for participants who answered "No voice," was the wake-up method. This participant had the characteristic of "not using voice information." The mean satisfaction scores were 0.25 (SD 1.22) for I1 (gazing), 1.50 (SD 0.90) for I2 (sign name), and 1.67 (SD 0.98) for I3 (waving). To examine the difference in the mean level of satisfaction among the three groups of the wake-up method, multiple comparisons using the Tukey's test were conducted. The results showed that the satisfaction level was significantly higher for I2 (signing name) and I3 (waving) than for I1 (gazing) ($p < 0.05$). For the "Identity," the main effect characteristics were the wake-up method and the feedback method within the participants who answered "Deaf." These participants were raised in a school for the

TABLE III
RESULTS OF ANALYSIS OF VARIANCE FOR TWO FACTORS OF SATISFACTION WITH WAKE-UP METHOD AND FEEDBACK PRESENTATION METHOD AS INDEPENDENT VARIABLES BY CHARACTERISTICS.

		Significance probability (p-value)		
Attributes	Characteristics	Wake-up (I1-I3)	Feedback (O1-O3)	Interaction
Whether using aids/cochlear implants	Yes (N=14)	0.63	0.81	0.67
	No (N=10)	4.45×10^{-3} **	0.22	0.98
Reading JSL level	JSL without voice (N=11)	1.20×10^{-3} **	0.12	0.82
	Pidgin JSL without voice (N=8)	0.80	0.68	0.92
	Pidgin JSL with voice (N=5)	0.71	0.50	0.83
Whether using voice in the conversation	Using voice (N=20)	0.38	0.63	0.61
	Not at all (N=4)	6.43×10^{-3} **	0.33	0.87
Identity	Deaf (N=8)	1.82×10^{-2} *	1.82×10^{-2} *	0.91
	Hard of hearing (N=6)	0.64	0.82	0.95
	Hearing impaired (N=9)	0.53	0.79	0.90

*: p < 0.05, **: p < 0.01

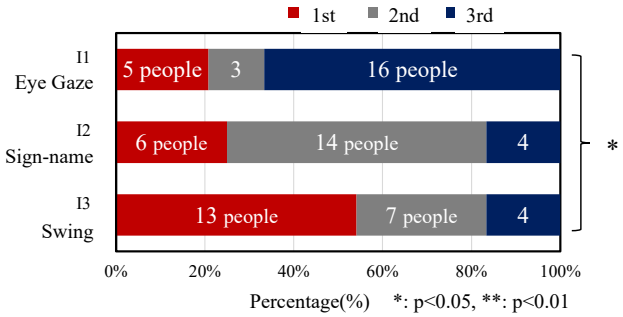


Figure 15. Ranking result of wake-up(N=24).

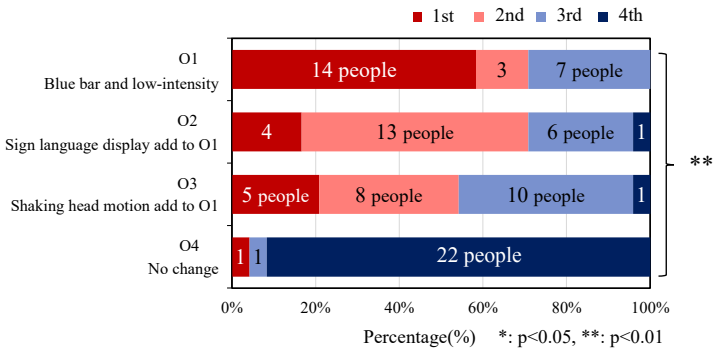


Figure 16. Ranking result of feedback (N=24).

Deaf, came from a Deaf family, and belonged to the Deaf community. For the wake-up method, the mean satisfaction level was 1.00 (SD 1.38) for I1 (gaze), 1.67 (SD 1.17) for I2 (sign name), and 1.96 (SD 1.00) for I3 (waving). Multiple comparisons of the Tukey's test were conducted to examine the differences in the mean satisfaction values among the three levels of the wake-up method. The results showed that the satisfaction level of I3 (waving) was significantly higher than that of I1 (gazing) ($p < 0.05$). For the feedback methods, the mean satisfaction values were 2.08 (SD 1.10) for O1 (blue bar and low-intensity), 1.42 (SD 1.10) for O2 (sign language added to O1), and 1.13 (SD 1.36) for O3 (head-shaking motion added to O1). Multiple comparisons of the Tukey's test were performed to examine the differences in the mean values of satisfaction among the three levels of feedback methods. The results showed that O1 (blue bar and low-intensity) was significantly more satisfactory than O3 (head-shaking motion added to O1) ($p < 0.05$).

3) Ranking: **Ranking results by overall participants** Figure 15 shows the results of the ranking done by the participants (N=24) for each wake-up condition (I1–I3). The mean rank was 2.46 (SD 0.82) for I1 (gazing), 1.92 (SD 0.64) for I2 (signing name), and 1.63 (SD 0.75) for I3 (waving). From these results, the Friedman test was used to investigate whether the mean rank varied with different wake-up methods, and the results showed a significant difference ($p < 0.05$).

For I1 (gazing), the participants who ranked it first mainly commented, "It was an easy calling action because it was hands-free," "It was the same feeling as using face recognition on a smartphone," "I was concerned that the system would recognize the user when they glanced at the system unintentionally. Therefore, it might be better to design the system to respond only after two seconds." In contrast, participants who ranked it third mainly commented, "I felt uncomfortable when I gazed at the screen when calling out," "I thought it would be better to call out using hands from the beginning if I gave a command in sign language afterward because I was unsure if the system would recognize me if I simply gazed at it. So it might be better to set a two-second rule to let the system react to gaze."

For I2 (sign name), the participants who ranked it first mainly commented, "I thought that the system would easily recognize the name-signing operation because it is a system-specific name," "Because it is a system-specific name and the system can easily recognize the name-signing operation, I felt comfortable with signing the name," "I feel closer to the system because it uses names." In contrast, the participants who ranked it third mainly commented, "I felt uncomfortable because I rarely call people by their names in my daily life," "I am not used to sign language, so I am not used to calling people by their sign name," "Compared to gazing and waving, I think that sign names are challenging to recognize."

For I3 (waving), the participants who ranked it first mainly said, "Because I can perform the action immediately," "Because it is the same action as calling out in daily conversation," "It is easy to call out even for the first time." In contrast, the participant who ranked it third commented, "Because the target of the name is not so clear compared to a sign name," "Because there was a possibility that my friends around me would misunderstand if I called out to the system with waving."

Figure 16 shows the results of the ranking done by the participants (N=24) for each feedback condition (O1–O4). The mean rankings were 1.71 (SD 0.91) in O1 (blue bar and low-intensity), 2.17 (SD 0.76) in O2 (sign language add to O1), 2.29 (SD 0.86) in O3 (shaking head motion add to O1), and 3.83 (SD 0.64) in O4 (no change) condition. (SD 0.64) in O4 (No change). We investigated the mean rank change with different feedback methods using the Friedman test from these results. We found a significant difference ($p < 0.01$).

For O1 (blue bar and low-intensity), the participants who ranked it first mainly commented, "I could immediately see that the system responded to my call," "The response was similar to Siri on my smartphone, so it was easy to get used to," "I felt that the other reactions were too much, and the blue bar was enough for me." In addition, there was a low evaluation comment, "It was too simple and difficult to understand compared to the sign language display and the head-shaking motion."

For O2 (sign language added to O1), the participants who ranked it first mainly commented, "I wanted the other person to respond in sign language because I spoke to them in sign

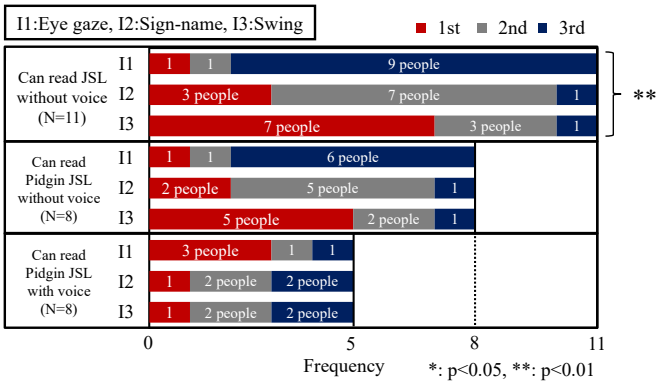


Figure 17. Ranking results for Wake-Up divided by JSL level attributes.

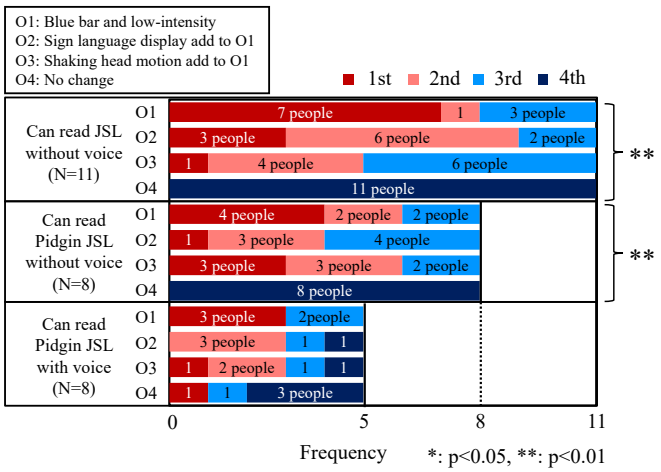


Figure 18. Ranking results for feedback divided by JSL level attributes.

language," "I felt relieved that the other person responded in sign language" "It was easy to understand their facial expressions and what they were asking." In contrast, there were comments such as "It would have been better if the system had used an avatar instead of a human," "I felt that it was sufficient to display text information instead of sign language," "It would be better if the system displayed the signer as an avatar instead of a human being, and if the signer was displayed from the default state, such as the home screen, instead of appearing only when I wake-up."

For O3 (shaking head motion added to O1), the participants who ranked it first mainly commented, "I could see that the system worked and responded clearly, and it gave me a cute impression," "It was straightforward to understand that the system responded to the shaking head." In contrast, there were comments on improving the head-shaking motion, such as, "I felt uncomfortable when the system turned to the outside. To wake-up the system without gazing, we would have preferred vertical to horizontal waving," "It was good in terms of visibility, but I was concerned that it would become difficult to see if more text information was added," "The head-shaking motion takes a little time, so the evaluation is lowered."

For O4 (no change), the participants who ranked it first commented, "No change was better because I can say what I want to say without pausing from the calling motion." In contrast, there were many comments with low evaluation, such as "When should I issue a command?," "I felt uneasy because I did not know when to give the command," "I want visible changes."

Ranking on sign language level characteristics Figure 17 summarizes the ranking data of the wake-up method for the sign language level attributes of the participants. We performed the Friedman test, and then we analyzed whether there was a significant difference in the level of satisfaction for each of the wake-up conditions (I1–I3) for each attribute. As a result, the only participant characteristic that showed a significant difference in satisfaction was the participants who answered "I can read JSL without voice" ($p < 0.01$). There was no significant difference in the level of satisfaction among the participants who answered "I can read Pidgin JSL without voice," "I can read Pidgin JSL with voice"($p > 0.05$).

Figure 17 summarizes the ranking data of the feedback method for the sign language level attributes of the participants. We performed the Friedman test, and then we analyzed whether there was a significant difference in the level of satisfaction for each feedback condition (O1–O4) for each attribute. As a result, the participant characteristics that showed significant differences in satisfaction were those who answered "I can read JSL without voice" ($p < 0.01$) and those who answered "I can read Pidgin JSL without voice" ($p < 0.01$). There was no significant difference in the level of satisfaction among the participants who answered "I can read Pidgin JSL with voice" ($p > 0.05$).

4) Participant behavior: **Time between the beginning and end of the wake-up action** We conducted multiple comparisons using the Tukey's test to see significant differences in the mean values of waving in each feedback condition (O1–O3). As a result, no significant difference was found ($p > 0.05$). Table IV summarizes the characteristics that reduced the time of waving compared to other attributes for each attribute of the participants. We analyzed four attributes: "whether the participant had hearing aids or cochlear implants," "JSL level,"

TABLE IV
PARTICIPANT CHARACTERISTICS FOR WHICH THE TIME OF WAVING WAS SIGNIFICANTLY SHORTER THAN THE OTHER CHARACTERISTICS.

Do a hearing aid/cochlear implant?	What is your JSL level?	Do you use your voice in conversation?	What is your identity?
Yes (N=14)	Can read JSL without voice * (N=11)	Yes (N=20)	Deaf * (N=8)
No (N=10)	Can read Pidgin JSL without voice (N=8)	No * (N=4)	Hard of Hearing (N=6)
	Can read Pidgin JSL with voice (N=5)		Hearing impaired (N=9)

* : Characteristics with significantly shorter swing time compared to other characteristics ($p < 0.05$)

TABLE V
COMPONENT RATIO AND DIFFERENCE IN START TIME FOR EACH OF EYE GAZE AND WAVING

First behavior	Component ratio %	Difference in start time (s) Mean (SD)	Min	Max
Eye Gaze	81.9	0.37 (0.35)	0.04	1.60
Waving	15.3	0.30 (0.32)	0.04	1.12
Same	2.8	n/a	n/a	n/a

TABLE VI
TIME OF SIMULTANEOUS EYE GAZE AND WAVING BEHAVIOR

Behavior	Time (s) Mean (SD)	Min	Max
Eye gaze and waving	1.07 (0.52)	0.20	2.46

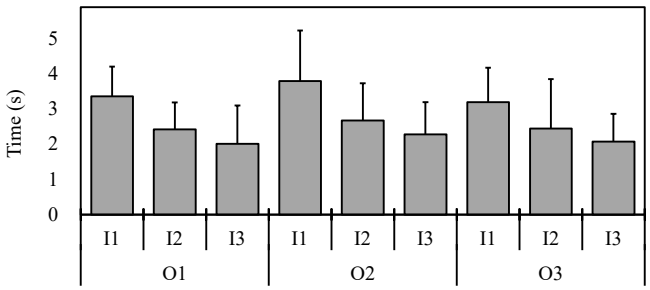


Figure 19. Mean of time from the end of wake-up to the start of command input for each 9 conditions.

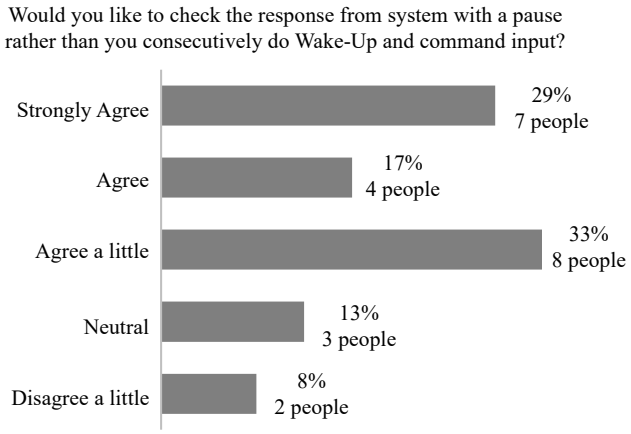


Figure 20. Necessity for feedback.

"whether participant voiced in conversation," and "Identity." To compare whether there was a significant difference between the levels, we performed Tukey's test of multiple comparisons for "JSL level" and "Identity" (three levels), and Welch's t-test for multiple comparisons for "whether the participants used hearing aids or cochlear implants" and "whether the participants used voice in conversation" (two levels). We found no significant difference in time for the former. For the attribute "JSL level," the time required for waving was 0.99 s (SD 0.35) for participants who answered "I can read JSL without voice" and 0.99 s (SD 0.35) for participants who answered "I can read Pidgin JSL with voice." We observed that the time taken for the waving was significantly shorter at higher sign language levels ($p < 0.01$). For the attribute "Using voice in the conversation," the time for waving was 1.17 s (SD 0.52) for participants who answered "Not at all" and 0.87 s (SD 0.41) for those who answered "Using voice." The time required for waving was significantly shorter for those who did not use their voices ($p < 0.05$). For the "Identity" attribute, the waving time was 0.93 s (SD 0.35) for participants who answered "Deaf" and 1.26 s (SD 0.61) for participants who answered "hearing-impaired." The time was significantly shorter for participants with a "Deaf" identity than for those with a "hearing-impaired" identity ($p < 0.01$).

Eye gaze in waving Table V shows the ratio of gaze initiation and waving first, the time from gaze initiation to the beginning of the waving motion, and the time from starting a waving gesture to beginning to gaze at the display. For the remaining 2.8% of the data, the difference between the starting times of the two behaviors were within 0.02 s. Table VI shows the times when both gazing and waving were performed.

We conducted multiple comparisons of the Tukey's test determine whether the difference in feedback methods affected the time spent performing both gazing and waving (O1–O3). The results showed that there were no significant differences between the two methods.

The time from the end of the wake-up to the beginning of command input Figure 19 shows the mean values and standard deviations of the time from the end of the wake-up to the start of command input for each of the nine conditions combining the wake-up method and the feedback method. We conducted a two-way ANOVA with the wake-up method (I1–I3) and feedback presentation method (O1–O3) as factors. The results showed that the main effect for the time of the wake-up method was significant ($p < 0.01$), but the main effect for the time of the feedback method was not significant ($p > 0.05$). In addition, the interaction of time for each of the wake-up and feedback methods was not significant ($p > 0.05$). For the wake-up method for which the main effect of time was significant, the mean time values were 3.45 s (SD 1.28) in I1 (eye gaze), 2.52 s (SD 1.20) in I2 (sign name), and 2.12 s (SD 1.20) in I3 (waving). To examine the difference in the mean satisfaction values among the three levels of the wake-up method, we conducted multiple comparisons using the Tukey's test. I3 (waving) was significantly shorter ($p < 0.01$) than I2 (sign name) ($p < 0.05$).

5) Subjective Evaluation: **Necessity for feedback presentation** Figure 20 shows the results of whether the participant preferred to check the response from the system with a pause rather than consecutively performing wake-up and command input. Participants who answered "1. strongly Agree" mainly said, "I can say what I want to say without anxiety if I can check system's response," "I was concerned that the system would have difficulty recognizing my sign language if I did not pause to check the system's response," "When I use sign

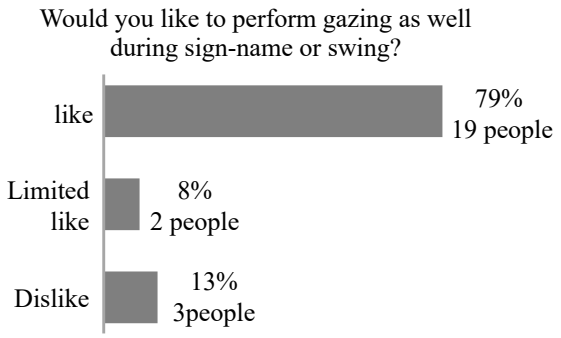


Figure 21. Necessity for gazing during sign name and waving.

language in my daily life, it is natural for me to pause every time I speak, and I would like to do so even if the other party is a system."

Participants who answered "2. agree" mainly commented, "I would like to input commands after I have confirmed that the system responds to my wake-up gesture," "I wanted to check the system's response, and I was concerned that if I didn't pause, it would be difficult for the system to recognize my sign language," "I don't think the pause is so necessary between hearing people. However, I want to be sure that the machine will respond to me before I say what I want."

Participants who answered "3. agree a little" mainly commented, "With voice control devices such as Alexa, I think I can talk without pauses. However, I would prefer to give the wake-up command and see the response from the system, as well as feedback before issuing further commands," "When I am cooking, I don't think I need a pause," "I believe that the system's recognizing of my sign language would be more stable if there was a pause. I thought that if the reading accuracy was as good as a human's, it might not need a pause."

Participants who answered "4. neutral" mainly commented, "It depends on the accuracy of the system's sign language recognition. If the system can recognize the sign language properly, it is not necessary to pause," "I am busy, and I want to send commands immediately after wake-up, considering situations where I have to multi-task."

Participants who answered "5. disagree a little" mainly commented, "Even in interpersonal communication, there is no time between the wake-up and the request. Rather, as the system can see me from the beginning, it should be able to notice my call immediately," "I am impatient, and I wanted to say what I wanted to say right after I woke up the system."

Necessity for gazing when performing wake-up with the hands Figure 21 shows the results of the question as to whether the participants preferred to perform gazing as well during wake-up movements using their hands, such as sign name and waving. Participants who answered "1. like" mainly commented, "I am concerned about the possibility of being mistakenly recognized when I wake-up the system without gazing at it me when talking with others, so I would like to gaze at the system as well," "I'm not used to calling out to people without looking at them because, in interpersonal communication in daily life, we call out with our eyes," "I tend to look at people with my eyes when I talk to them, so I feel safer when I add gazing," "By adding gazing, I can bidirectionally know that the target of the call is the system," "I want to add gazing so that I can check the system's response."

Participants who answered "2. limited like" mainly commented, "I thought that we don't need to gaze at the target of the call because the target of the call can be clearly identified just by signing the name, which is a system-specific way of calling," "If waving is used to call out to the people around me, I would like to gaze at them as well to clarify the target of the call," "If you are busy at work, you may be able to call out without gazing."

Participants who answered "3. dislike" mainly commented, "I don't think gazing is necessary with the possibility of signing the name and waving. If the system can read my sign language, then there is no problem," "I don't feel like calling out while looking at the screen when I envision using the system after coming home. I don't want to be tied to gazing at the screen. I want to look at the screen when the system responds, not to wake it up."

D. Discussion

1) RQ4:Need for feedback: Usability is defined in ISO 9241-11, a standard of the International Organization for Standardization, as "the degree of effectiveness, efficiency, and user satisfaction in which a given user uses a product to achieve a specified goal under specified conditions of use" [56]. From Figure 12, it may be observed that the participants' satisfaction was significantly higher when the system provided feedback, indicating that the usability of the system in terms of "degree of user satisfaction" was improved. However, Figure 13 shows that the time between the call to the system and the command's input was significantly shorter when the system did not provide feedback than when it did do so, indicating that the usability improved in terms of "efficiency". From Figure 16, it may be observed that the ranking of satisfaction was lower when feedback was not provided. In addition, as there were many comments about feeling uneasy, safety could not be ensured, and therefore, the usability decreased in terms of "effectiveness". This suggests that it is more important for usability improvements to obtain a proper response from the system than to speed up the command input in the conversation between a DHH user and the system. In addition, Figure 20 shows that DHH users tended to prefer to check the response from the system by leaving some time between the wake-up action and the command input. These results suggest that the system must provide DHH users with feedback for their wake-up actions.

2) RQ5:Optimal wake-up method: From Figure 14 and Figure 15, it is clear all DHH users preferred wake-up with waving to eye gaze or signing a name. From Section IV-C3, a comment was made by a participant, "waving is often used for calling out in interpersonal communication, so it was easy." This is because it has been reported that Deaf people mainly

wave their hands when they try to contact a physically distant person [29]. From these findings, we can say that the wake-up of the system partner showed behaviors similar to Deaf people's conversations. From Section IV-C3, the participants commented that the waving was immediately transferable. From Figure 19, the time between the end of the wake-up action and the start of the command input was significantly shorter for the waving than for the other wake-up conditions. This suggests that waving is practical for a smooth transition to the command input.

Table V shows that 97.2% of the time, the participants were gazing when they performed the waving. Figure 21 shows that even when participants performed wake-up actions using their hands, such as sign name and waving, they tended to perform them in conjunction with gazing. These results suggest that using gaze and waving is an optimal wake-up method for DHH users.

From Table III, participants whose physical characteristics and identity were "Deaf", such as "not relying on auditory information," "high sign language level," and "not using audio information," were significantly more satisfied with waving than with the other wake-up methods. Figure 17 shows that they greatly preferred waving in the satisfaction ranking. In addition, from Table IV, the time required for waving was significantly shorter for participants whose physical characteristics and identities were "Deaf", such as "high sign language level" and "not relying on auditory information," than for the participants with other characteristics. Deaf people have been learning sign language since they were young and are used to communicating with their hands, including sign language, because they belong to the Deaf community. It has been reported that DHH individuals have better visual senses and are more aware of their surroundings than ordinary people [57]. Therefore, if the person you call out to is Deaf, the other person is more likely to notice even if the waving time is short. This suggests that participants who have the physical characteristics of the Deaf are more likely to be accustomed to such situations and therefore prefer hand movements such as waving.

In contrast, from Table III, participants whose physical characteristics and identity were Hard of hearing (hearing impaired), such as "relying on auditory information," "low sign language level," and "using audio information," did not show a preference for waving compared to other wake-up methods. From Table IV, the time required for waving was significantly longer than that of participants whose physical characteristics and identity were "Deaf". This suggests that DHH participants are resistant to using waving as a wake-up movement because such movements are time-consuming. Figure 17 shows that three of the five participants (66% of the total) with a low sign language level preferred gazing. This suggests that hearing-impaired users may prefer not to use their hands, and simply use gaze instead.

3) RQ6:Optimal feedback method: From Figure 14, there was no way to present the system's feedback that resulted in a significantly higher level of satisfaction among all DHH users. However, it is clear from Figure 16 that for all DHH users, the blue bar display was preferred over other feedback.

Regarding the characteristics of DHH users, Figure 18 shows that participants whose identity was "Deaf" were significantly more satisfied with the blue bar display only than with the system's head-shaking motion, and participants with a higher sign language level especially preferred the blue bar display only. In Section IV-C3, there were comments from participants such as "Blue bar display alone is enough," so we suggest that the short time it took for the system to show a response improved the usability of the system.

However, the blue bar display was asymmetrical in that the participants' input method was sign language, but the output method was text or the blue bar display. The blue bar display is asymmetrical from the text and blue bar display. Figure 18 shows that no participants with a low sign language level ranked the sign language display first. Still, many participants with a high sign language level ranked the sign language display as their first or second preference. As for why the satisfaction level of the sign language display was lower than that of the blue bar-only display, based on the participants' comments on the sign language display in the Section IV-C3, it is possible that improving the specifications of the sign language display, such as "displaying the signer as an avatar" or "displaying the signer from the default state, such as the home screen, so that it responds to the user's input," could improve the satisfaction level. These results suggest that the feedback of the system needs to be further studied.

V. DESIGN GUIDELINES FOR A CONVERSATIONAL NATURAL USER INTERFACE FOR THE DEAF AND HARD OF HEARING USERS

From Section III and Section IV, based on the experimental investigation of the research questions, Figure 22 shows the style of the process that DHH users should realize when conversing with a conversational NUI. We propose five design guidelines for conversational NUI for DHH users.

1. Allow wake-up to be performed on a directed gaze
   For DHH users, it is natural to gaze at the system and then enter commands (RQ3). However, it should be noted that the preferred wake-up method differed depending on user characteristics (RQ5). For users who are Deaf, waving can be used as an option to personalize the system. In this case, the designer can use the Table VI data as a reference to create a recognition system for waving and eye gaze.
2. Provide feedback for wake-up
   DHH users prefer confirmation before entering commands to the system after wake-up (RQ4). To make DHH users feel secure, the system should provide feedback such as a sign language display.
3. Command input should be in sign language
   There is some research on alternative input methods to speech for DHH command input, such as sign language [7] [8]. As DHH users are interested in interacting with the system using sign language [9],

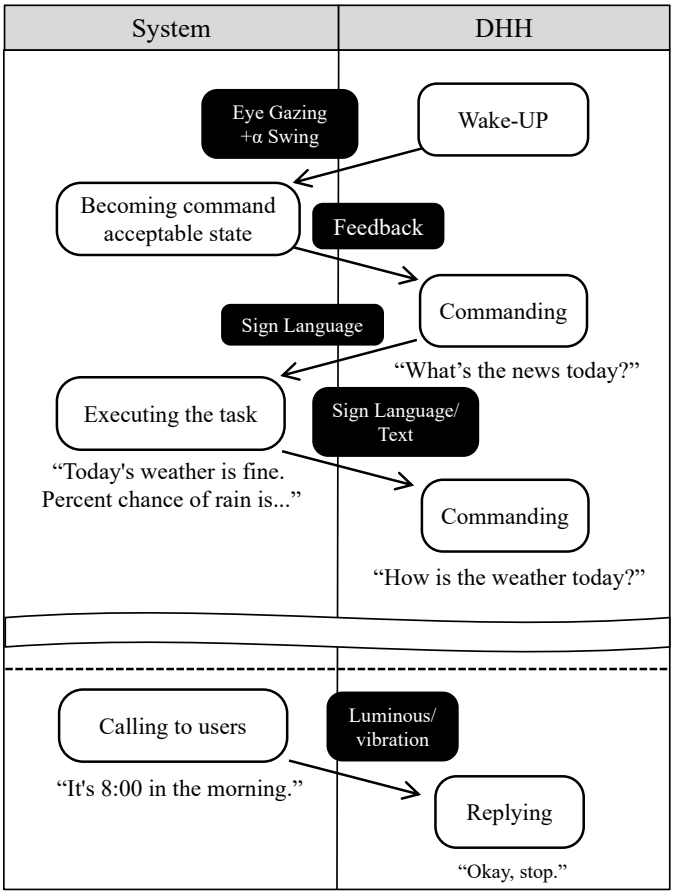


Figure 22. Activity of the conversation process between the DHH user and the system.

they should be able to provide input to the system using sign language.

4. System outputs sign language and text

   Text output is the best form of interaction with the system in multi-tasking. For users who mainly use sign language input, sign language is the best output mode for the system (RQ2). However, the sign language output may not be satisfactory for users who are not accustomed to sign language, so the display method needs to be personalized.

5. Use illumination or vibration to notify the user

   There are times when the DHH user is not looking at the system, so they are called by it and do not notice. To avoid this, the system outputs a luminous notification (RQ1). In addition, the use of vibrations for notifications was considered following the guidelines for mobile applications (applications that run directly on devices such as smartphones and tablets), which were developed with the DHH user experience in mind [58]. Therefore, we can expect vibration-based notification methods in conversational NUIs.

VI. LIMITATIONS AND FUTURE WORK

In this study, the age range of the participants was low (20-24), with all participants being university students, and the sample size was small, with a maximum of 24 participants. Therefore, it was impossible to investigate the preferences and behavior patterns of a wide range of age groups. It has been reported that there are differences in preferences between younger and older people for CUIs equipped with AI assistants [59]. Therefore, evaluation experiments should be conducted with a more diverse range of people in future research.

In addition, 92% of the participants in Experiment 1 and 83% of the participants in Experiment 2 had little or no experience using VUIs. In other words, as the participants were not used to the system, the system's behavior was not very predictable, and their evaluation of the system may change with more regular use. Therefore, an evaluation experiment should be conducted after participants are entirely accustomed to using the system. In addition, issues and comfort levels in daily life should be explored, and fieldwork over extended periods of time should be considered.

In this study, we incorporated a luminous notification as a means of responding to DHH users. However, some participants commented, "I think it would be easier to notice if there was a notification method using vibration as well as light." In the future, we intended to conduct an experiment that includes a vibration notification. In addition, because we placed the system in front of the participants in this experiment, we need to find a way to make them aware of the notifications from behind.

Avatar display functions must also be extended for sign language display and the user face-tracking function for system rotation motion. It is also necessary to investigate usability by participants.

In addition, we examined the system's sign language/text display method and the method of presenting feedback for wake-up, but we did not review feedback for failure to recognize a user's command. When interacting with the system using sign language, experiments should be conducted on possible innovations such as making the signer on the system appear to be asking a question.

When designing a CUI, conventional approaches often consider only a one-time task [52]. However, human conversation rarely involve only a single exchange. Therefore, in the future, it is desirable to design CUIs that allow conversations to continue further. Here, a study reported that in Deaf interpersonal communication, whether a conversation is interrupted is mainly due to the end of mutual gaze [29]. Therefore, it is expected that the gazing modality can be used to determine how to end a conversation. Therefore, it is necessary to conduct experiments on conversational termination method in the future as well.

VII. CONCLUSION

In this study, we have proposed design guidelines for conversational NUIs for DHH users. We have investigated

optimal accessibility methods for DHH users at each step of a conversation with VUIs. To this end, we conducted two experiments. In Experiment 1, we asked for responses from DHH users (N=12) to investigate whether a sign language conversation system using luminous notification and gazing could improve usability. In Experiment 2, we collected responses from DHH users (N=24) to investigate optimal wake-up and feedback presentation method.

The main empirical contributions of this work are summarized as follows.

(1) We have provided evidence showing that output with luminous notifications increased DHH users' satisfaction.
(2) We have also demonstrated the necessity of sign language/text output for DHH users.
(3) We have provided evidence that gaze can serve as a natural wake-up method for DHH users, but some users prefer waving.
(4) We have also provided evidence of a high need for DHH users to be provided feedback on wake-up.
(5) Finally, we have developed guidelines for conversational NUIs best suited for DHH users.

This study serves as a design guideline for future conversational NUIs to improve accessibility for DHH users.

References

[1] T. Kato, A. Shitara, N. Kato, and Y. Shiraishi, "Sign Language Conversational User Interfaces Using Luminous Notification and Eye Gaze for the Deaf and Hard of Hearing," Proceedings of ACHI'21: The 14th International Conference on Advances in Computer-Human Interactions, pp. 30–36, Nice, France, 2021.

[2] R. Byron, N. Clifford, "The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places," Bibliovault OAI Repository, the University of Chicago Press, pp. 18–36, 1996.

[3] H. Candello et al., "CUI@CHI: Mapping Grand Challenges for the Conversational User Interface Community," CHI EA '20: In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–8, New York, USA, April 2020.

[4] TECH DRIVERS, (February 15, 2022), "Making sense of Google CEO Sundar Pichai's plan to move every direction at once," https://www.cnbc.com/2017/05/18/google-ceo-sundar-pichai-machine-learning-big-data.html

[5] J. P. Bigham, R Kushalnagar, T. K. Huang, J. P. Flores, S. Savage, "On How Deaf People Might Use Speech to Control Devices," ASSETS '17: In Proceedings of ASSETS '17: The 19th International ACM SIGACCESS Conference on Computers and Accessibility, New York, USA, pp. 383–384, October 2017.

[6] A. Glasser, "Automatic Speech Recognition Services: Deaf and Hard-of-Hearing Usability," In Extended Abstracts of CHI EA '19: The 2019 CHI Conference on Human Factors in Computing Systems, No. SRC06, New York, USA, pp. 1–6, May 2019.

[7] G. Evan, K. Raja S., R. Jason, V. Christian, W. Brittany, "Accessibility of voice-activated agents for people who are deaf or hard of hearing," In Proceedings of CSUN '19: The34th Annual Assistive Technology Conference Scientific/Research, Vol. 7, pp. 144–156, San Diego, 2019.

[8] W. Gilmore et al., "Alexa, Can You See Me?" Making Individual Personal Assistants for the Home Accessible to Deaf Consumers," Proceedings of ASSETS '19: The 35th Annual Assistive Technology Conference Scientific/Research, Vol. 8, pp. 16–31, San Diego, USA, 2020.

[9] A. Glasser, V. Mande, M. Huenerfauth, "Understanding deaf and hard-of-hearing users' interest in sign-language interaction with personal-assistant devices," In Proceedings of W4A '21: The 18th International Web for All Conference, Association for Computing Machinery, pp. 1–11, New York, USA, 2021.

[10] A. Glasser, V. Mande, M. Huenerfauth, "Accessibility for Deaf and Hard of Hearing Users: Sign Language Conversational User Interfaces," In Proceedings of CUI '20: The 2nd Conference on Conversational User Interfaces, New York, USA, No. 55, pp. 1–3, July 2020.

[11] D. Bragg et.al., "Sign Language Interfaces: Discussing the Field's Biggest Challenges," In Extended Abstracts of CHI EA '20: The 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–5, New York, USA, April 2020.

[12] D. Bragg et.al., "Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective," In Proceedings of ASSETS '19: The 21st International ACM SIGACCESS Conference on Computers and Accessibility, pp. 16–31, New York, USA, October 2019.

[13] Nielsen Norman Group, (February 15, 2022), "10 Usability Heuristics for User Interface Design," https://www.nngroup.com/articles/ten-usability-heuristics/

[14] J. Nielsen, "Enhancing the explanatory power of usability heuristics," In Proceedings of CHI '94: The SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, pp. 152–158, New York, USA, 1994.

[15] D. Norman, "The design of everyday things," Basic Books, 1988.

[16] B. Shneiderman, C. Plaisant, "Designing the User Interface: Strategies for Effective Human-Computer Interaction," Addison Wesley, 2010.

[17] C. Murad, C. Munteanu, L. Clark, B. R. Cowan, "Design guidelines for hands-free speech interaction," In Proceedings of MobileHCI '18: The 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct, Association for Computing Machinery, pp. 269–276, New York, USA, 2018.

[18] Alexa, (February 15, 2022), "Design Principles," https://developer.amazon.com/en-US/alexa/alexa-haus/design-principles

[19] Google Assistant, (February 15, 2022), "Conversational Actions, Guides Design guidelines," https://developers.google.com/assistant/interactivecanvas/design

[20] C. Murad, C. Munteanu, "'I don't know what you're talking about, HALexa': the case for voice user interface guidelines," In Proceedings of CUI '19: The 1st International Conference on Conversational User Interfaces, Association for Computing Machinery, No. 9, pp. 1–3, New York, USA, 2019.

[21] J. Berke, (February 15, 2022), "Assistive Listening Devices for the Deaf and HOH," https://www.verywellhealth.com/assistive-listening-devices-1046105

[22] A. Matsuda, M. Sugaya, H. Nakamura, " Luminous device for the deaf and hard of hearing people. In Proceedings of the second international conference on Human-agent interaction," In Proceedings of HAI '14: The second international conference on Human-agent interaction, pp. 201–204, October 2014.

[23] Amazon, (February 15, 2022), "Hearing Communicate and stay connected with Alexa," https://www.amazon.com/b/ref=ods_afe_hop_hp?node=21213721011

[24] F. Bentley et al., "Understanding the Long-Term Use of Smart Speaker Assistants," In Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 2, No. 3, pp 1–24, September 2018.

[25] M. Marschark et al., "Benefits of Sign Language Interpreting and Text Alternatives for Deaf Students' Classroom Learning," The Journal of Deaf Studies and Deaf Education, Vol. 11, pp. 421–437, Fall 2006.

[26] S. Deepika, R. Rangasayee, "The Combined Effect of Captioning and Sign Language in Understanding Television Content in Deaf," Journal of Communication Disorders, Deaf Studies & Hearing Aids, Vol. 6, No. 1, pp. 1–7, 2018.

[27] M, Debevc, P. Kosec, A. Holzinger, "Improving multimodal web accessibility for deaf people: sign language interpreter module," Multimed Tools Appl Vol. 54, pp. 181–199, April 2010.

[28] SignGenius, (February 15, 2022), "Do's & Don'ts - Getting Attention in the Deaf Community," https://www.signgenius.com/info-do's&don'ts.shtml

[29] U. Bartnikowska, "Significance of touch and eye contact in the Polish Deaf community during conversations in Polish Sign Language: ethnographic observations," Hrvatska Revija za Rehabilitacijska Istraživanja, Vol. 53, pp. 175–185, 2017.

[30] V. Mande, A. Glasser, B. Dingman, M Huenerfauth, "Deaf Users' Preferences Among wake-up Approaches during Sign-Language Interaction with Personal Assistant Devices," In Extended Abstracts of CHI EA '21: The 2021 CHI Conference on Human Factors in Computing Systems, Vol. 370, pp 1–6, May 2021.

[31] C. Heath, K. Nicholls, Body Movement and Speech in Medical Interaction, Studies in Emotion and Social Interaction, Cambridge University Press, 1986.

[32] A. M. Lieberman, "Attention-getting skills of deaf children using American Sign Language in a preschool classroom," Applied psycholinguistics, Vol. 36, No. 4, pp. 855–873, July 2016.

[33] National University Corporation, Tsukuba University of Technology, (February 15, 2022), "About," https://www.tsukuba-tech.ac.jp/english/index.html

[34] I. Lopatovska et al., "Talk to me: Exploring user interactions with the Amazon Alexa," Journal of Librarianship and Information Science, Vol. 51, No. 4, pp. 984–997, 2019.

[35] S. Ahire, A. Priegnitz, O. Önbas, M, Rohs, W. Nejdl, "How Compatible is Alexa with Dual Tasking? — Towards Intelligent Personal Assistants for Dual-Task Situations," In Proceedings of HAI '21: The 9th International Conference on Human-Agent Interaction, Association for Computing Machinery, pp. 103–111, New York, USA, 2021.

[36] Canalys, (February 15, 2022), "Global smart speaker market 2021 forecast," https://canalys.com/newsroom/canalys-global-smart-speaker-market-2021-forecast

[37] National Institutes of Health (US); Biological Sciences Curriculum Study, "NIH Curriculum Supplement Series," Bethesda (MD): Information about Hearing, Communication, and Understanding, 2007.

[38] Pixel Phone Help, (February 15, 2022), "Control your Pixel without touching it," https://support.google.com/pixelphone/answer/9517454?hl=en

[39] F. Donna, "The Use of waving as Self-Generated Cues for Recall of Verbally Associated Targets," The American journal of psychology, Vol. 115, No. 1, pp. 1–20, 2002.

[40] F. Donna, R. E. Guttentag, "The Effects of Restricting waving Production on Lexical Retrieval and Free Recall," The American Journal of Psychology, Vol. 115, No. 1, pp. 1–20, 2002.

[41] M. Henschke, T. Gedeon, R. Jones, "Touchless Gestural Interaction with Wizard-of-Oz: Analysing User Behaviour," In Proceedings of OzCHI '15: The Annual Meeting of the Australian Special Interest Group for Computer Human Interaction, Association for Computing Machinery, pp. 207–211, New York, USA, 2015.

[42] J. Schwarz, C. C. Marais, T Leyvand, S. E. Hudson, J. Mankoff, "Combining body pose, gaze, and gesture to determine intention to interact in vision-based interfaces," In Proceedings of CHI '14: The SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, pp. 3443–3452, New York, USA, 2014.

[43] W. Yee, "Potential Limitations of Multi-touch Gesture Vocabulary: Differentiation, Adoption, Fatigue," In Proceedings of the 13th International Conference on Human-Computer Interaction, Part II: Novel Interaction Methods and Techniques. Springer-Verlag, pp. 291–300, Berlin, Heidelberg, July 2009.

[44] R. Cui, H. Liu, C. Zhang, "A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training," in IEEE Transactions on Multimedia, Vol. 21, No. 7, pp. 1880–1891, July 2019.

[45] N. Crook, (February 15, 2022), "Wizard of Oz testing – a method of testing a system that does not yet exist," https://www.simpleusability.com/inspiration/2018/08/wizard-of-oz-testing-a-method-of-testing-a-system-that-does-not-yet-exist/

[46] N. Fraser, N. Gilbert, "Simulating speech systems. Computer Speech & Language," Vol. 5, pp. 81–99, January 1991.

[47] A. Pradhan, K. Mehta, L. Findlater, ""Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities," In Proceedings of CHI '18: The 2018 CHI Conference on Human Factors in Computing Systems, No. 459, pp. 1–13, New York, USA, 2018.

[48] J. Peirce, J. R. Gray, "Simpson, S. et al. PsychoPy2: Experiments in behavior made easy," Behavior Research Methods, Vol. 51, pp. 195–203, February 2019.

[49] The Language Archive, (June 5, 2021), "ELAN," https://archive.mpi.nl/tla/elan

[50] J. Brooke, "SUS: A quick and dirty usability scale," Usability Eval. Ind.. 189. pp. 1–7. November 1995.

[51] D. Bavelier et al., "Visual attention to the periphery is enhanced in congenitally deaf individuals," The Journal of neuroscience : the official journal of the Society for Neuroscience, Vol. 20, No. 17, pp. 1–8. September 2000.

[52] C. Pearl, "Designing Voice User Interfaces: Principles of Conversational Experiences," O'Reilly Media, 2017.

[53] Developer documentationamazon alexa, (February 15, 2022), "Alexa Design Guide. Be Avaiable," https://developer.amazon.com/en-GB/docs/alexa/alexa-design/available.html

[54] Amazon, (February 15, 2022), "What Do the Lights on Your Echo Device Mean?," https://www.amazon.com/gp/help/customer/display.html?nodeId=GKLDRFT7FP4FZE56

[55] Amazon, (February 15, 2022), "Echo Show 10 (3rd Gen) — HD smart display with motion and Alexa — Charcoal," https://www.amazon.com/echo-show-10/dp/B07VHZ41L8

[56] Online Browsing Platform(OBP), (February 15, 2022), "ISO 9241-11:2018(en)" https://www.iso.org/obp/ui/\#iso:std:iso:9241:-210:ed-2:v1:en

[57] D. Bavelier, W.G. Matthew, P. C. Hauser, "Do deaf individuals see better?," Trends in Cognitive Sciences, Vol. 10, No. 11, pp. 512–518, 2006.

[58] R. P. Schefer, M. S. Bezerra, L. A. M. Zaina, "Supporting the Development of Social Networking Mobile Apps for Deaf Users: Guidelines Based on User Experience Issues," In Proceedings of DSAI 2018: The 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, Association for Computing Machinery, pp. 278—285, New York, USA, 2018.

[59] D. Gollasch, G. Weber, "Age-Related Differences in Preferences for Using Voice Assistants," In Proceedings of MuC '21: In Mensch und Computer 2021, Association for Computing Machinery, pp. 156–167, New York, USA, 2021.

An Extended Study of the Correlation of Cognitive Complexity-related Code Measures

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
email:luigi.lavazza@uninsubria.it

***Abstract*—Several measures have been proposed to represent various characteristics of code, such as size, complexity, cohesion, coupling, etc. These measures are deemed interesting because the internal characteristics they measure (which are not interesting *per se*) are believed to be correlated with external software qualities (like reliability, maintainability, etc.) that are definitely interesting for developers or users. Although many measures have been proposed for software code, new measures are continuously proposed. However, before starting using a new measure, we would like to ascertain that it is actually useful and that it provides some improvement with respect to well established measures that have been in use for a long time and whose merits have been widely evaluated. In 2018, a new code measure, named "Cognitive Complexity" was proposed. According to the proposers, this measure should correlate to code understandability much better than traditional code measures, such as McCabe Complexity, for instance. However, hardly any experimentation proved whether the "Cognitive Complexity" measure is better than other measures or not. Actually, it was not even verified whether the new measure provides different knowledge concerning code with respect to traditional measures. In this paper, we aim at evaluating experimentally to what extent the new measure is correlated with traditional measures. To this end, we measured the code from a set of open-source Java projects and derived models of "Cognitive Complexity" based on the traditional code measures yielded by a state-of-the-art code measurement tool. We found that fairly accurate models of "Cognitive Complexity" can be obtained using just a few traditional code measures. In this sense, the "Cognitive Complexity" measure does not appear to provide additional knowledge with respect to previously proposed measures.**

Keywords–Cognitive complexity; software code measures; McCabe complexity; cyclomatic complexity; Halstead measures; static code measures

I. Introduction

In [1], the correlation between "Cognitive Complexity," a measure proposed with the aim of representing the complexity of understanding code [2], and "traditional" measures was studied.

In fact, many measures of the internal characteristics of code, such as size, complexity, cohesion or coupling, have been proposed in the past (for instance, Chidamber and Kemerer proposed a suite of metrics that are suitable for representing the characteristics of object-oriented code [3]) and new ones are continuously proposed. However, code measures are of little interest *per se*, since they address internal properties of software. In general, developers, managers and users are more interested in external software qualities, like faultiness or maintainability. Therefore, it is necessary that internal property measures are correlated to some external property of interest. Such correlation makes it possible, among other things, to predict interesting external qualities, which are unknown, based on measures of internal code properties, which can be easily collected.

In 2018, a new code measure was proposed with the aim of representing the complexity of understanding code [2]. The measure is a code measure, which accounts exclusively for internal code properties. However, according to the author, it is expected to be strictly correlated with code understandability, which is an external code property. This measure is named "Cognitive Complexity," however, in the remainder of this paper we shall refer to this measure as "CoCo," to avoid confusion with the *concept* of cognitive complexity, i.e., the external property that CoCo is expected to measure.

Some initial work has been done to evaluate whether CoCo is actually correlated with code understandability [4]: preliminary results do not support the claim that CoCo is better correlated to code understandability than previously proposed measures.

At any rate, whatever the goal that a new code measure is supposed to help achieving, the new measure should provide some "knowledge" that existing code measures are not able to capture. If a new measure is so strongly correlated with other measures that the latters can be used to predict the new measure with good accuracy, it is unlikely that the new measure actually conveys any new knowledge.

CoCo is receiving some attention, probably because it is provided by `SonarQube`, which is a quite popular tool. Therefore, it is time to look for evidence that CoCo provides additional knowledge with respect to well established code measures. To this end, the following two research questions were addressed by a previous paper [1]:

- RQ1 How strongly is CoCo correlated with each of the code measures that are commonly used in software development?
- RQ2 Is it possible to build models that predict the value of CoCo based on the values of commonly used code measures? If so, how accurate are the predictions that can be achieved?

These research questions were addressed by analyzing the code from nine open source Java projects. It was found that CoCo appears strongly correlated to McCabe's complexity and slightly less strongly correlated to several other code measures. Several regression models of CoCo as a function of traditional measures were also found. Based on these findings, it was concluded that—at least for the considered software projects—CoCo does not appear to convey additional information with respect to traditional measures.

In this paper, we first report in some detail the work described in [1], then we look for further evidence that may confirm (or challenge) previous findings. To this end,

1) We selected two large open source Java projects, whose code we used in the following activities.
2) We evaluated the correlation between CoCo and traditional measures in the two selected open source Java projects.
3) We used two models found in [1] to predict the CoCo of the two projects' methods. We then evaluated the accuracy of the prediction.
4) Finally, we used machine learning techniques to verify whether it is possible to estimate CoCo based on traditional measures with an even greater accuracy than via regression.

The paper is structured as follows. Section II provides some background, by introducing CoCo and describing the traditional code measures used in this study. Section III describes the empirical study that was carried out to answer the research questions. Section IV discusses the results obtained by the original study [1] and answers the research questions. Section V describes a second empirical study, which provides further evidence that confirms previous findings. Section VI explores whether it is possible to uncover even stronger relationships by means of machine learning, or if the usage of ML just confirms the previous findings obtained via ordinary least squares regression. Section VII discusses the threats to the validity of the study. Section VIII accounts for related work. Finally, in Section IX some conclusions are drawn, and future work is outlined.

II. Code Measures

In this paper we deal with measures of the internal attributes of code. Internal attributes of code can be measured by looking at code alone, without considering software qualities (like faultiness, robustness, maintainability, etc.) that are externally perceivable.

Several measures for internal software attributes (e.g., size, structural complexity, cohesion, coupling) were proposed [5] to quantify the properties of software modules. These measures are interesting because they concern code properties that are believed to affect external software qualities (like faultiness or maintainability), which are interesting for developers and users.

Since CoCo is computed at the method level, in what follows, we consider only measures at the same granularity level, i.e., measures that are applicable to methods.

A. "Traditional" Code Measures

Since the first high-level programming languages were introduced, several measures were proposed, to represent the possibly relevant characteristics of code. For instance, the Lines Of Code (LOC) measure the size of a software module, while McCabe Complexity (also known as Cyclomatic Complexity) [6] was proposed to represent the "complexity" of code, with the idea that high levels of complexity characterize code that is difficult to test and maintain. The object-oriented measures by Chidamber and Kemerer [3] were proposed to recognize poor software design: for instance, modules with high levels of coupling are supposed to be associated with difficult maintenance.

In this paper, we are interested in evaluating the correlation between CoCo and traditional measures. Since CoCo is defined at the method level, here we consider only traditional measures addressing methods; measures defined to represent the properties of classes or other code structures are ignored.

We used SourceMeter [7] to collect code measures. The collected method-level measures are listed in Table I.

Here we provide just a brief description of the collected measures; readers can find complete specifications and additional information in the documentation of SourceMeter.

The measures listed in Table I include Halstead measures [8], several maintainability indexes, including the original one [9], McCabe complexity, measures of the nesting level (i.e., how deeply are code control structures included in each other), logical lines of code (which are counted excluding blank lines, comment-only lines, etc.).

TABLE I
THE MEASURES COLLECTED VIA SOURCEMETER.

Metric name	Abbreviation
Halstead Calculated Program Length	HCPL
Halstead Difficulty	HDIF
Halstead Effort	HEFF
Halstead Number of Delivered Bugs	HNDB
Halstead Program Length	HPL
Halstead Program Vocabulary	HPV
Halstead Time Required to Program	HTRP
Halstead Volume	HVOL
Maintainability Index (Microsoft version)	MIMS
Maintainability Index (Original version)	MI
Maintainability Index (SEI version)	MISEI
Maintainability Index (SourceMeter version)	MISM
McCabe's Cyclomatic Complexity	McCC
Nesting Level	NL
Nesting Level Else-If	NLE
Logical Lines of Code	LLOC
Number of Statements	NOS

B. The "Cognitive Complexity" Measure

In 2017, SonarSource introduced Cognitive Complexity [2] as a new measure for the understandability of any given piece of code. This new measure was named "Cognitive Complexity" because its authors assumed that the measure was suitable to represent the cognitive complexity of understanding code. To this end, CoCo was proposed with the aim *"to remedy*

Cyclomatic Complexity's shortcomings and produce a measurement that more accurately reflects the relative difficulty of understanding, and therefore of maintaining methods, classes, and applications" [2].

Rather than a direct measure, CoCo is an indicator, which takes into account several aspects of code. Like McCabe's complexity, it takes into account decision points (conditional statements, loops, switch statements, etc.), but, unlike McCabe's complexity, CoCo gives them a weight equal to their nesting level plus 1. So, for instance, in the following code fragment

```
void firstMethod() {
  if (condition1)
    for (int i = 0; i < 10; i++)
      while (condition2) { x+=a[i]; }
}
```

the `if` statement at nesting level 0 has weight 1, the `for` statement at nesting level 1 has weight 2, and the `while` statement at nesting level 2 has weight 3; accordingly CoCo= $1+2+3=6$. The same code has McCabe complexity = 4 (3 decision points plus one).

Consider instead the following code fragment, in which the control structures are not nested.

```
void secondMethod() {
  if (condition1) { x=0; }
  for (int i = 0; i < 10; i++) { x+=2*i; }
  while (condition2) { x=x/2; }
}
```

This code has CoCo = 3, while its McCabe complexity is still 4. It is thus apparent that nested structures increase CoCo, while they have no effect on McCabe complexity.

CoCo also accounts for Boolean predicates (while McCabe's complexity does not): a Boolean predicate contributes to CoCo depending on the number of its sub-sequences of logical operators. For instance, consider the following code fragment, where `a, b, c, d, e, f` are Boolean variables

```
void thirdMethod() {
  if (a && b && c || d || e && f) { ... }
}
```

Predicate `a && b && c || d || e && f` contains three sub-sequences with the same logical operators, i.e., `a && b && c`, `c || d || e`, and `e && f`, so it adds 3 to the value of CoCo.

Other aspects of code contribute to increment CoCo, but they are much less frequent than those described above. For a complete description of CoCo, see the definition [2].

III. The Empirical Study

The empirical study involved a set of open-source Java programs. The Java code was measured, and the collected data were analyzed via well consolidated statistical methods. The dataset is described in Section III-A, while the measurement and analysis methods are described in Section III-B. The results we obtained are reported in Section III-C.

A. The Dataset

The code to be analyzed within the study was a convenience sample: data whose code was already available from previous studies concerning completely different topics was used. In practice, this amount to a random choice.

The projects that supplied the code for the study are listed in Table II, where some descriptive statistics for the most relevant measures are also given (for space reasons, statistics are given only for a subset of representative measures). Methods having CoCo=0 (i.e., with no decidion points, no complex boolean expressions, etc.) or NOS=0 (i.e., having no statements) are clearly uninteresting, therefore their data were excluded, so Table II does not account for such methods. Overall, the initial dataset included data from 13,922 methods. The dataset is available on demand for replication purposes.

B. The Method

The first phase of the study consisted in measuring the code. We used SourceMeter to obtain the traditional measures listed in Table I, and a self-constructed tool to measure CoCo. The data from the two tools were joined, thus obtaining a single dataset with 8,214 data points.

The second step consisted in selecting the data for the study. We excluded from the study all the methods having CoCo < 5, since those methods would bias the results, because of 'built-in' relationships. For instance a piece of code having CoCo = 0 also has McCabe complexity = 1; similarly, CoCo = 1 implies that McCabe complexity = 2 for all but a few very peculiar cases, etc. In addition, low-complexity methods are of little interest: CoCo is meant to represent the complexity of understanding code, and CoCo is less than 5 for methods that are so simple that understanding them is hardly an issue. Therefore, by excluding only methods having CoCo < 5 we are sure to exclude only 'non-interesting' code.

We also excluded methods having CoCo > 50, because our dataset contains too few methods having CoCo > 50 to support reliable statistical analysis. Besides, CoCo > 50 indicates exceedingly complex methods; in practice, it is hardly useful knowing if, say, CoCo = 60 or CoCo = 70, just like it is hardly useful knowing that McCabe's complexity is 60 or 70. In these cases, we just have "too complex" methods.

After removing the exceedingly simple or complex methods, we got a dataset including 3,610 data points, definitely enough to perform significant statistical analysis. In this dataset the mean value of CoCo is 12, while the median is 9.

The third step consisted in performing statistical analysis. We started by studying the correlation between CoCo and each one of the other code measures. Since the data are not normally distributed, we used non-parametric tests, namely we computed Kendall's rank correlation coefficient τ [10] and Spearman's rank correlation coefficient ρ [11]. Since the correlation analysis gave encouraging results, we proceeded to evaluate correlations via both linear and non-linear correlation

TABLE II
DESCRIPTIVE STATISTICS OF THE DATASETS.

Project	num. methods	Measure	mean	st.dev.	median	min	max
hibernate	2532	CoCo	3.1	4.3	2.0	1	79
		HPV	32.3	17.1	28.0	0	211
		MI	100.3	14.7	102.2	0	135
		McCC	3.3	2.4	2.0	1	33
		NLE	1.3	0.8	1.0	0	7
		LLOC	15.2	12.3	12.0	3	201
jcaptcha	317	CoCo	3.3	4.0	2.0	1	34
		HPV	35.0	18.4	29.0	10	120
		MI	100.3	14.0	102.5	56	132
		McCC	3.5	2.2	3.0	2	18
		NLE	1.3	0.8	1.0	0	5
		LLOC	14.6	10.6	11.0	3	80
jjwt	205	CoCo	4.0	7.2	2.0	1	84
		HPV	30.6	22.9	28.0	0	280
		MI	101.7	20.6	104.0	0	135
		McCC	4.3	4.6	3.0	2	46
		NLE	1.3	0.8	1.0	0	4
		LLOC	13.5	14.9	11.0	3	169
json_ iterator	379	CoCo	5.6	8.7	3.0	1	73
		HPV	38.3	21.1	32.0	14	145
		MI	96.4	15.3	99.0	45	131
		McCC	4.6	3.9	3.0	1	28
		NLE	1.6	1.0	1.0	0	7
		LLOC	18.0	15.1	13.0	3	110
JSON-java	260	CoCo	5.7	15.8	2.0	1	203
		HPV	41.0	36.9	31.5	11	413
		MI	95.7	18.2	97.4	32	133
		McCC	5.0	5.8	3.0	2	50
		NLE	1.5	1.1	1.0	0	7
		LLOC	21.5	26.5	13.0	3	255
log4j	798	CoCo	4.6	6.4	2.0	1	61
		HPV	36.6	21.4	30.0	8	163
		MI	98.1	15.2	100.4	44	135
		McCC	4.1	3.4	3.0	1	34
		NLE	1.6	1.0	1.0	0	8
		LLOC	16.9	13.4	12.0	3	115
netty-socketio	136	CoCo	4.4	5.5	3.0	1	37
		HPV	33.7	20.0	28.0	0	122
		MI	97.7	20.8	101.4	0	132
		McCC	4.1	2.8	3.0	1	19
		NLE	1.6	0.9	1.0	0	5
		LLOC	15.0	12.3	11.0	3	84
pdfbox	3587	CoCo	5.2	8.2	2.0	1	118
		HPV	39.3	25.7	32.0	0	326
		MI	93.7	17.2	96.4	0	128
		McCC	4.5	4.5	3.0	1	58
		NLE	1.6	1.1	1.0	0	10
		LLOC	22.3	21.8	15.0	3	330
jasper reports	6415	CoCo	5.6	10.1	3.0	1	186
		HPV	38.7	28.5	31.0	0	740
		MI	93.4	18.1	96.5	0	132
		McCC	4.9	5.6	3.0	1	117
		NLE	1.6	1.1	1.0	0	10
		LLOC	23.5	26.0	15.0	3	383

analysis. Namely, we performed ordinary least squares (OLS) linear regression analysis and OLS regression analysis after log-log transformation of data. In both cases, we identified and excluded outliers based on Cook's distance [12].

In all the performed analysis, we considered the results significant at the usual $\alpha = 0.05$ level.

C. Results of the Study

The results of Kendall's and Spearman's correlation tests are given in Table III. All the reported results are statistically significant, with p-values well below 0.001.

After the evaluation of correlations between CoCo and other measures, we proceeded to building regression models. We obtained 65 statistically significant models after log-log transformation of measures. Table IV provides a summary of the most accurate models we found. For each model, the adjusted R^2 determination coefficient is given (obtained after excluding outliers). We also give a few indicators of the accuracy of the models (computed including outliers): MAR is the mean of absolute residuals (i.e., the average absolute prediction error), MMRE is the mean magnitude of relative errors, while MdMRE is the median magnitude of relative errors. MMRE and MdMRE are considered biased indicators: we report them here only as a complement to MAR, which we considered the indicator of accuracy to be taken into account [13].

Note that in addition to the measures listed in Table I, we used also MCC/LLOC, i.e., McCabe's complexity density.

TABLE III
RESULTS OF CORRELATION TEST.

Measure	τ	ρ
HCPL	0.45	0.62
HDIF	0.38	0.52
HEFF	0.47	0.63
HNDB	0.47	0.63
HPL	0.50	0.67
HPV	0.46	0.62
HTRP	0.47	0.63
HVOL	0.50	0.66
MI	−0.56	−0.73
MIMS	−0.56	−0.73
MISEI	−0.41	−0.57
MISM	−0.41	−0.57
McCC	0.71	0.85
NL	0.50	0.61
NLE	0.50	0.60
LLOC	0.55	0.72
NOS	0.52	0.68

IV. DISCUSSION OF THE RESULTS FROM THE EMPIRICAL STUDY

The results of the correlation tests given in Table III show that CoCo is correlated with all the traditional code measures we considered. Specifically, CoCo is strongly correlated with McCabe's complexity: this is quite noticeable, considering that CoCo was proposed to improve McCabe's complexity.

We can thus answer RQ1 as follows:
Our study shows medium to strong correlations between CoCo and each of the commonly used code measures that we considered. Specifically, CoCo appears most strongly correlated with McCabe's complexity.

The results given in Table IV let us answer RQ2 as follows: Our study shows that it possible to build models that predict the value of CoCo based on commonly used measures, as well as using Halstead measures and maintainability indexes. Many of the obtained models feature quite good accuracy.

Noticeably, the independent variables that support the most accurate models are McCabe's complexity, the nesting level and the number of logical lines of code. This is hardly surprising, given that elements of MCC and NLE are used in the definition of CoCo. As to LLOC, it is clear that the longer the code, the more decision points it contains (on average),

TABLE IV
MODELS FOUND.

Measures	adjusted R^2	MAR	MMRE	MdMRE
MI, NL	0.81	3.60	0.28	0.20
MIMS, NL	0.81	3.60	0.28	0.20
NLE, LLOC	0.79	3.08	0.25	0.20
HCPL, MI, NLE	0.84	2.96	0.24	0.18
HCPL, MIMS, NLE	0.84	2.96	0.24	0.18
HCPL, NLE, LLOC	0.81	3.04	0.25	0.20
HDIF, MI, NL	0.82	3.65	0.28	0.19
HDIF, MI, NLE	0.84	2.96	0.24	0.19
HDIF, MIMS, NL	0.82	3.65	0.28	0.19
HDIF, MIMS, NLE	0.84	2.96	0.24	0.19
HEFF, MI, NL	0.82	3.72	0.28	0.20
HEFF, MI, NLE	0.84	3.01	0.24	0.19
HEFF, MIMS, NL	0.82	3.72	0.28	0.20
HEFF, MIMS, NLE	0.84	3.01	0.24	0.19
HNDB, MI, NL	0.82	3.72	0.28	0.20
HNDB, MI, NLE	0.84	3.01	0.24	0.19
HNDB, MIMS, NL	0.82	3.72	0.28	0.20
HNDB, MIMS, NLE	0.84	3.01	0.24	0.19
HPL, MI, NLE	0.84	3.03	0.24	0.19
HPL, MIMS, NLE	0.84	3.03	0.24	0.19
HPL, NLE, LLOC	0.82	3.03	0.25	0.20
HPV, MI, NL	0.82	3.77	0.28	0.20
HPV, MI, NLE	0.84	2.95	0.24	0.18
HPV, MIMS, NL	0.82	3.77	0.28	0.20
HPV, MIMS, NLE	0.84	2.95	0.24	0.18
HTRP, MI, NL	0.82	3.72	0.28	0.20
HTRP, MI, NLE	0.84	3.01	0.24	0.19
HTRP, MIMS, NL	0.82	3.72	0.28	0.20
HTRP, MIMS, NLE	0.84	3.01	0.24	0.19
HVOL, MI, NLE	0.84	3.04	0.24	0.19
HVOL, MIMS, NLE	0.84	3.04	0.24	0.19
HVOL, NLE, LLOC	0.82	3.03	0.25	0.20
MI, MIMS, NLE	0.81	3.59	0.26	0.19
MI, NL, NLE	0.81	2.89	0.23	0.18
MI, NLE, LLOC	0.83	3.25	0.25	0.19
MIMS, NL, NLE	0.81	2.89	0.23	0.18
MIMS, NLE, LLOC	0.83	3.25	0.25	0.19
McCC, NLE, LLOC	0.95	1.77	0.15	0.11
McCC, NLE, MCC/LLOC	0.95	1.77	0.15	0.11
NL, NLE, LLOC	0.78	2.99	0.24	0.20
NLE, LLOC, MCC/LLOC	0.95	1.77	0.15	0.11

hence we can expect also LLOC to contribute to CoCo. In fact, the relationship between CoCo and lines of code was already observed [14].

In conclusion, our study shows that CoCo does not seem to convey more knowledge than sets of properly chosen traditional code measures, like MCC, NLE and LLOC.

V. EXPERIMENTAL VERIFICATION OF FORMER RESULTS

In this section we report the results of a second empirical study, which provides further evidence that confirms the findings given above.

A. The verification dataset

To verify the results from [1] we selected two large open source Java projects, namely `ant 1.10.12` and `tomcat 10.0.0-M10`. The descriptive statistics of the two projects' code are given in Table V.

B. Verifying correlation of CoCo with traditional measures

The first verification activity we carried out consisted in testing the correlation between CoCo and traditional measures. To this end, we computed Kendall's rank correlation coefficient τ [10] and Spearman's rank correlation coefficient ρ [11], as was done in [1].

TABLE V
DESCRIPTIVE STATISTICS OF THE NEW DATASETS.

Project	num. methods	Measure	Mean	st.dev.	Median	Min	Max
ant	3505	CoCo	4.73	7.60	2	1	107
		HPV	34.22	23.07	27	0	188
		MI	101.20	17.91	104	0	136
		McCC	4.40	4.30	3	1	53
		NLE	1.44	1.06	1	0	9
		LLOC	16.19	16.30	11	3	162
tomcat	8050	CoCo	6.47	14.44	3	1	413
		HPV	39.64	30.42	31	0	651
		MI	96.75	18.76	100	-14	150
		McCC	5.12	6.77	3	1	154
		NLE	1.69	1.12	1	0	13
		LLOC	19.78	24.59	12	1	612

TABLE VI
RESULTS OF THE NEW CORRELATION TESTS.

Measure	τ	ρ
HCPL	0.47	0.63
HDIF	0.41	0.56
HEFF	0.49	0.66
HNDB	0.49	0.66
HPL	0.52	0.69
HPV	0.47	0.64
HTRP	0.49	0.66
HVOL	0.52	0.69
MI	−0.58	−0.75
MIMS	−0.58	−0.75
MISEI	−0.43	−0.58
MISM	−0.43	−0.58
McCC	0.71	0.86
NL	0.50	0.61
NLE	0.50	0.61
LLOC	0.58	0.75
NOS	0.55	0.72

The results we obtained are given in Table VI. In all cases, the p-value was less than 10^{-3}.

The results in Table VI fully confirm the previous results reported in Table III. Specifically, in `ant` and `tomcat`, the correlation between CoCo and traditional measures appears just a bit stronger. However, the differences in both τ and ρ are so small that they fully confirm the reliability of the correlation coefficients given in [1].

C. Evaluation of the accuracy of CoCo models

As we mentioned in Section III-C, several models of CoCo as a function of traditional measures were found. A selection is given in Table IV, where accuracy indications obtained via classical 10-time 10-fold cross-validation are also reported.

We can now use the new dataset containing measures from `ant` and `tomcat` to test the accuracy of those models. If we achieve accurate predictions, that means that the models obtained from the original dataset represent a fairly general relationship between CoCo and traditional measures.

The results from the original analysis suggest that the following two models are the most accurate ones:

$$CoCo = 0.6408 McCC^{0.8105} NLE^{0.6404} LLOC^{0.1552} \quad (1)$$

$$CoCo = 0.6515 \left(\frac{McCC}{LLOC}\right)^{0.8440} NLE^{0.6392} LLOC^{0.9651} \quad (2)$$

So, we used models (1) and (2) to estimate the CoCo of `ant` and `tomcats` methods, based on McCC, NLE and LLOC of those applications' methods.

When using model (1) we obtained the absolute error illustrated by the boxplot in Figure 1 (outliers not shown). The blue diamond represents the MAR.

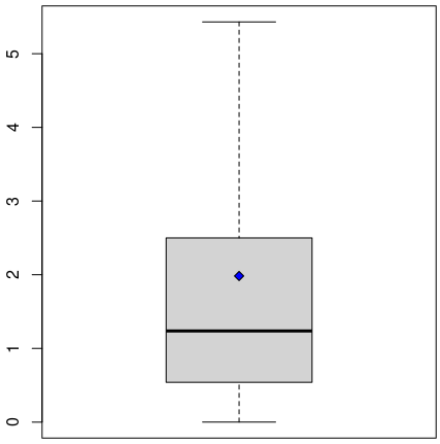


Fig. 1. Distribution of absolute errors of estimates obtained via (1), without outliers.

Figure 2 shows the distribution of absolute relative errors (including outliers). It can be seen that the greatest majority of estimates is within 5% of the actual CoCo.

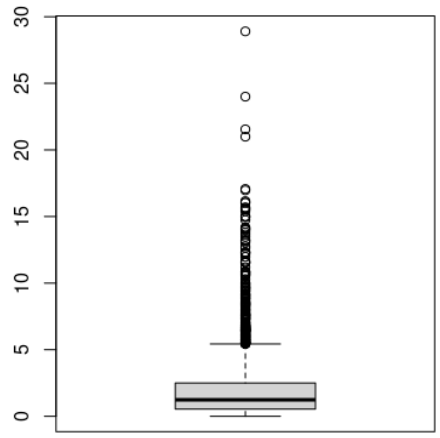


Fig. 2. Distribution of absolute relative errors of estimates obtained via (1), with outliers.

Figure 3 compares actual CoCo values with estimates obtained via (1). The blue straight line represents the perfect prediction.

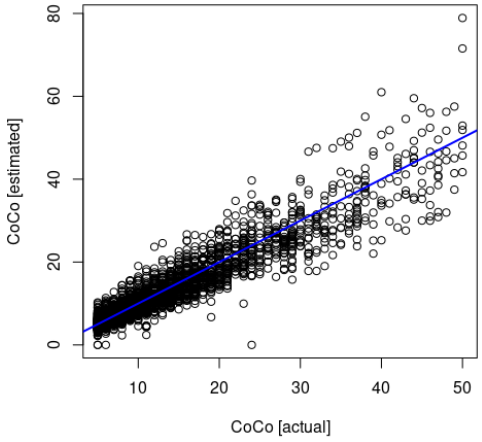


Fig. 3. Comparison of actual CoCo with estimates obtained via (1).

When using model (2) we obtained the absolute error illustrated by the boxplot in Figure 5 (outliers not shown).

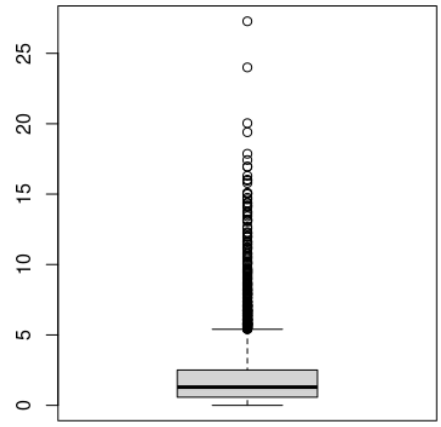


Fig. 4. Distribution of absolute relative errors of estimates obtained via (2), with outliers.

Figure 4 shows the distribution of absolute relative errors (including outliers). It can be seen that also in this case the greatest majority of estimates is within 5% of the actual CoCo.

Figure 6 compares actual CoCo values with estimates obtained via (2).

Overall, the evaluation of models (1) and (2) via the `ant` and `tomcat` dataset yielded results extremely close to those obtained with the 10-times 10-fold cross validation, as shown in Table VII, where column "10-times 10-fold Xval" reports the data already given in Table IV, concerning the accuracy evaluated on the original dataset, while column "`ant` and `tomcat` prediction" provides the accuracy indicators for the predictions obtained applying (1) and (2) to the `ant` and `tomcat` dataset.

In conclusion, the models of CoCo confirm that there is a strong correlation between CoCo and traditional measures, and that it is possible to get a quite accurate estimate of CoCo based on models that have traditional measures as independent variables.

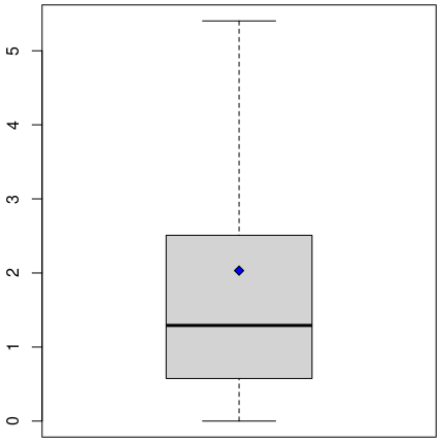


Fig. 5. Distribution of absolute errors of estimates obtained via (2), without outliers.

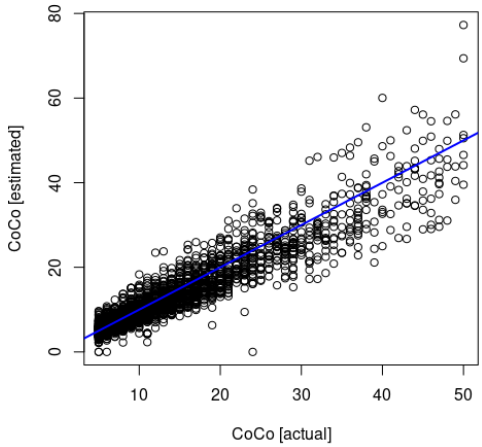


Fig. 6. Comparison of actual CoCo with estimates obtained via (2).

VI. CoCo estimation using machine learning

Previous sections showed that CoCo does not seem to add much information with respect to traditional measures (especially McCabe complexity, NLE and logical LOC). In this section we explore whether it is possible to uncover even stronger relationships by means of machine learning (ML), or if the usage of ML just confirms the previous findings obtained via ordinary least squares (OLS) regression.

We proceeded through the following steps:

1) We used the original dataset to build a model of CoCo vs. McCC, NLE and LLOC. To this end, we used Support Vector Regression (SVR) with radial kernel. The computations were carried out via the R language and programming environment [15], using the `e1071` library. Parameters γ, ϵ and cost were trained to minimize the MAR (Mean Absolute Residual) via repeated application of a 5-fold cross validation sampling method (to this end, the `tuning` function of the `e1071` library was used).
2) We used the resulting model to estimate CoCo for each method of `ant` and `tomcat`.
3) We evaluated the accuracy of the obtained estimates, and compared then with the estimates obtained via OLS regression (as described in Section V-C).

Figure 7 shows the distribution of estimation errors (without outliers). It is easy to see that the greatest majority of estimates is quite correct; namely over 50% of the estimation errors are in [-1, +1] range.

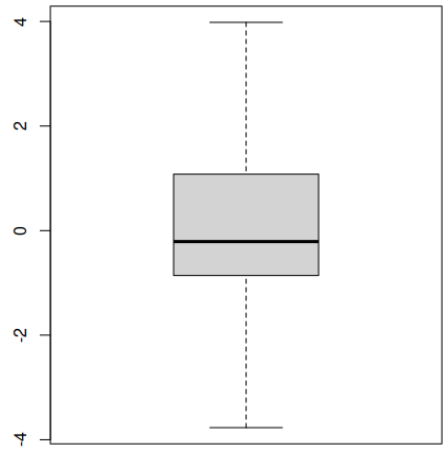


Fig. 7. Distribution of errors of estimates obtained via ML (no outliers).

Figure 8 shows the distribution of absolute estimation errors (without outliers). The blue diamond represents the MAR. It is easy to see that over 75% of the estimation errors have magnitude less than 2.5.

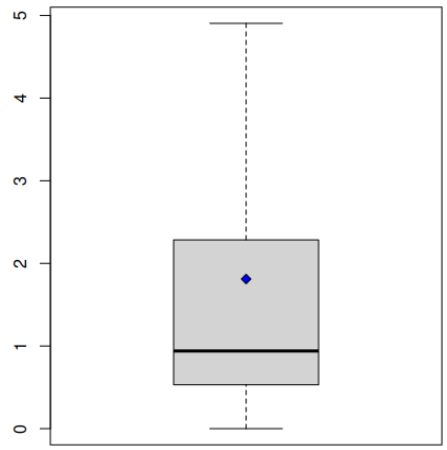


Fig. 8. Distribution of absolute errors of estimates obtained via ML (no outliers).

TABLE VII
ACCURACY OF MODELS (1) AND (2).

	10-times 10-fold Xval			`ant` and `tomcat` prediction		
model	MAR	MMRE	MdMRE	MAR	MMRE	MdMRE
(1)	1.77	0.15	0.11	1.98	0.16	0.13
(2)	1.77	0.15	0.11	2.03	0.16	0.14

Figure 9 compares the estimates with the actual CoCo values. In can be seen that most estimates are very close to the corresponding actual values. However, in a few cases, the estimates are relatively far from the actual value.

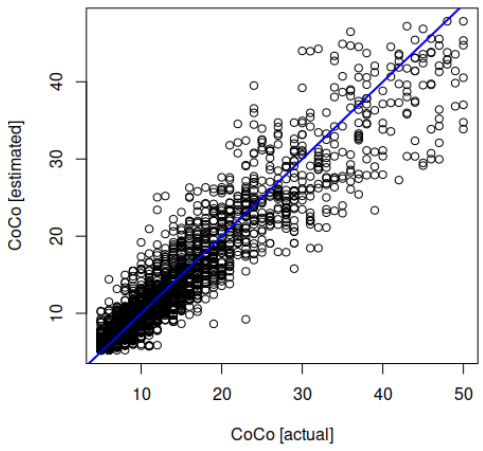


Fig. 9. Comparison of actual CoCo measures and estimates.

The summary of accuracy indicators is:

- MAR= 1.8
- MMRE= 0.145
- MdMRE= 0.115

So, the accuracy of ML models is quite close to the accuracy of OLS models, as is apparent by comparing the values above with those in Table VII.

VII. Threats to Validity

Concerning the application of traditional measures, we used a state-of-the-art tool (SourceMeter), which is widely used and mature, therefore we do not see any threat on this side. CoCo was measured using an ad-hoc tool that was built based on the specifications of CoCo [2]. This tool was thoroughly tested using `SonarQube` [16] as a reference, therefore we are reasonably sure that it provides correct measures. However, when joining the data from SourceMeter with the data from our tool, we were not able to always match methods identifiers, because the two tools reported slightly different descriptions of methods' names, parameters, etc. We just dropped the methods' data for which no sure match could be found: in this way, we lost less than 2% of the measures. Since the lost measures depend on characteristics that have nothing to do with the properties of code being measured, they can be considered a random subset, which can hardly affect the outcomes of the study.

Concerning the external validity of the study, as with most empirical studies in the Software Engineering area, we cannot be completely sure about the generalizability of results. However, the dataset used was large enough, and the selected software projects represent a reasonable variety of application types. In addition, the verification performed using a new dataset fully confirmed the original results [1]. Also the usage of SVR to evaluate the correlation of CoCo with traditional measure confirmed the original findings. We can thus conclude that the presented results appear reliable and reasonably general.

VIII. Related Work

Campbell performed an investigation of the developers' reaction to the introduction of CoCo in the measurement and analysis tool `SonarCloud` [17]. In an analysis of 22 open-source projects, she assessed whether a development team "accepted" the measure, based on whether they fixed code areas indicated by the tool as characterized by high CoCo. Around 77% of developers expressed acceptance of the measure.

An objective validation of the CoCo measure was performed by Muñoz Barón et al. [4]. They retrieved data sets from published studies that measured the understandability of source code from the perspective of human developers. They collected the data concerning various aspects of understandability, as well as the code snippets used in the experiments. They used `SonarQube` [16] to obtain the CoCo measure for each source code snippet. Then, they computed the correlation of CoCo with the measures of various aspects of understandability. Muñoz Barón et al. computed the correlation between CoCo and various aspects of understandability for each of the 10 experiments reported in the selected papers, as well as a summary obtained via meta-analysis. Muñoz Barón et al. concluded that CoCo correlates moderately with some of the considered understandability aspects.

The paper mentioned above dealt with evaluating the effectiveness of CoCo (a measure of internal code properties) as an indicator of understandability (an external code property). To our knowledge, nobody performed an analysis dealing with how internal code properties only are correlated with CoCo.

Nonetheless, CoCo has been used in some evaluations. CoCo is provided by `SonarQube` [16] together with many other measures and indicators, so some researchers that used `SonarQube` to collect code measures ended up using CoCo together with other measures. Among the papers that have used CoCo are the following.

Kozik et al. [14] developed a framework for analyzing software quality dependence on code measures and other data. Using the framework they found that CoCo affects the analyzability and adaptability of code.

Papadopoulos et al. [18] investigated the interrelation between design time quality metrics and runtime quality metrics, such as cache misses, memory accesses, memory footprint and CPU cycles. Papadopoulos et al. observed a trade-off between performance/energy consumption and cognitive complexity. However, having used CoCo as the only design time quality metric, it is unknown whether the same kind of trade-off would be observed with respect to other design-time metrics, like McCabe's complexity, for instance. Our study suggests that this doubt is well funded, i.e., a trade-off involving performance/energy consumption and design-time metrics like McCabe's complexity could very well exist.

Crespo et al. [19] used both the Cognitive complexity rate (defined as CoCo/LOC) and the Cyclomatic complexity rate (defined as McCabe complexity/LOC) as part of an assessment strategy concerning technical debt in an educational context. They found that the Cognitive complexity rate and the Cyclomatic complexity rate provide the same results, or lack of results, actually. Given the strong correlation that we observed between CoCo and McCabe's complexity, the result by Crespo et al. is not surprising.

IX. Conclusions

The "Cognitive Complexity" measure (CoCo throughout the paper) was introduced with the aim of improving the ability to detect code that is difficult to understand and maintain [2]. Rather than a direct measure, CoCo is an indicator, whose definition accounts for a few characteristics of source code. Among these characteristics are the number of decision points (e.g., if, for, while and switch statements) and the level of nesting of control statements.

When CoCo was proposed, no evaluations were published concerning the relationship between CoCo and traditional measures that directly address the aforementioned characteristics of code. In this paper, we have reported about empirical studies aiming at evaluating the correlation between CoCo and several traditional measures, including those addressing the same characteristics of code taken into account by CoCo. To this end, we measured a few open source projects' code, obtaining the measures of 3,610 methods. We then performed statistical analysis using both correlation tests (namely, Kendall's and Spearman's rank correlation coefficients), regression analysis and machine learning.

We found that CoCo appears strongly correlated to McCabe's complexity and slightly less strongly correlated to several other code measures. We found several regression models of CoCo as a function of traditional measures. Not surprisingly, one of the most accurate models involves McCabe's complexity, NLE (Nesting Level Else-If) and LLOC (the number of logical lines of code) as independent variables. Considering that the most accurate models have MAR=1.7, while the mean CoCo is 12, we may conclude that—at least for the considered software projects—CoCo does not appear to convey additional information with respect to traditional measures.

Cross-dataset validation confirmed the initial results, as did the models obtained using Support Vector Regression.

In conclusion, the study reported here casts the doubt that CoCo does not provide appreciable new knowledge with respect to the measures of code that are traditionally associated with the notion of complexity.

Concerning future work, it can be noticed that the work reported here concerns exclusively relationships among internal measures. It could be interesting to evaluate how well the studied internal measures (CoCo and traditional complexity and size measure) correlate with external qualities. Specifically, we plan to repeat previous studies [20], [21] using CoCo together with (or alternatively to) other code measures.

Acknowledgment

The work reported here was partly supported by Fondo per la Ricerca di Ateneo, Università degli Studi dell'Insubria. The author thanks Anatoliy Roshka for developing the tool that was used to measure CoCo.

References

[1] L. Lavazza, "An Empirical Study of the Correlation of Cognitive Complexity-related Code Measures," in Proceedings of The Sixteenth International Conference on Software Engineering Advances – ICSEA, 2021.

[2] G. A. Campbell, "Cognitive complexity - a new way of measuring understandability," https://www.sonarsource.com/docs/CognitiveComplexity.pdf, 2018, [Online; accessed 7-September-2021].

[3] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," IEEE Transactions on software engineering, vol. 20, no. 6, 1994, pp. 476–493.

[4] M. M. Barón, M. Wyrich, and S. Wagner, "An empirical validation of cognitive complexity as a measure of source code understandability," in Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020, pp. 1–12.

[5] N. Fenton and J. Bieman, Software metrics: a rigorous and practical approach. CRC press, 2014.

[6] T. J. McCabe, "A complexity measure," IEEE Transactions on software Engineering, no. 4, 1976, pp. 308–320.

[7] "SourceMeter," https://www.sourcemeter.com/, [Online; accessed 7-September-2021].

[8] M. H. Halstead, Elements of software science. Elsevier North-Holland, 1977.

[9] P. Oman and J. Hagemeister, "Metrics for assessing a software system's maintainability," in Proceedings Conference on Software Maintenance 1992. IEEE Computer Society, 1992, pp. 337–338.

[10] M. G. Kendall, "Rank and product-moment correlation," Biometrika, 1949, pp. 177–193.

[11] C. Spearman, "The proof and measurement of association between two things," The American journal of psychology, vol. 100, no. 3/4, 1987, pp. 441–471.

[12] R. D. Cook, "Detection of influential observation in linear regression," Technometrics.

[13] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," Information and Software Technology, vol. 54, no. 8, 2012, pp. 820–827.

[14] R. Kozik, M. Choraś, D. Puchalski, and R. Renk, "Q-rapids framework for advanced data analysis to improve rapid software development," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 5, 2019, pp. 1927–1936.

[15] R core team, "R: a language and environment for statistical computing," 2015.

[16] "SonarQube," https://www.sonarqube.org/, [Online; accessed 7-September-2021].

[17] G. A. Campbell, "Cognitive complexity: An overview and evaluation," in Proceedings of the 2018 International Conference on Technical Debt, 2018, pp. 57–58.

[18] L. Papadopoulos, C. Marantos, G. Digkas, A. Ampatzoglou, A. Chatzigeorgiou, and D. Soudris, "Interrelations between software quality metrics, performance and energy consumption in embedded applications," in Proceedings of the 21st International Workshop on software and compilers for embedded systems, 2018, pp. 62–65.

[19] Y. Crespo, A. Gonzalez-Escribano, and M. Piattini, "Carrot and stick approaches revisited when managing technical debt in an educational context," arXiv preprint arXiv:2104.08993, 2021.

[20] V. Del Bianco, L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, "An investigation of the users' perception of OSS quality," in IFIP International Conference on Open Source Systems. Springer, 2010, pp. 15–28.

[21] ——, "The QualiSPo approach to OSS product quality evaluation," in Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 23–28.

Continuous Information Processing Addressing Cisco's Pain Points by Enabling Real-Time Ad-Hoc Reporting Capability: An Energy Efficient Big Data Approach

Martin Zinner*, Wolfgang E. Nagel*
* Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden
Dresden, Germany
E-mail: {martin.zinner1,wolfgang.nagel}@tu-dresden.de

***Abstract*—Aggregation is a special type of association, in which objects are assembled/composed together to create a more complex object. Unfortunately, a considerable part of data aggregation during information processing in industry is still carried out in nightly batch mode, taking into account all its negative side effects. In particular, before starting aggregation, all data required for computation must be available and accessible. In contrast, our method termed Continuous Information Processing Methodology (CIPM), does not assume that the data to be aggregated is fully retrieved, the aggregation can be started as soon as the data collection is initiated. During the data collection process, partial aggregated values are calculated, such that, after the data collection phase has been completed, the final aggregated values are available for real-time ad-hoc evaluation. The existing aggregation methods apply only the usual aggregation functions such as sum(), avg(), max(), min(), count() as build in functions, on the contrary, the most common aggregation functions used in various field of industry and business can be easily adapted and used within CIPM. The major benefit of our method is the elimination of the extra time necessary for batch computation, as well as reduced and spread aggregation effort over the whole collection period and tightened and straightforward computational design strategies. To conclude, the CIPM supports a paradigm shift from more or less subjectively designed individualistic conceptions in software design and development towards objectively established optimal solutions.**

***Index Terms*—Continuous information processing; Continuous aggregation; Energy efficient computation; Real-time capability; Real-time capabilities; Data Analytics; Data processing; Stream processing; Batch processing; Business Intelligence; Ad-hoc reporting; Big Data.**

I. Introduction

Initially, within this section, the core of our aggregation theory is succinctly addressed, some definitions such as that of Big Data are tightened up and subsequently, the principal motivation of our paper, i.e., increased real-time requirements in the industry, is presented. Aggregation is a special type of association, in which objects are in general assembled/composed together to create a new and a more complex object. The aim of aggregating data is to create new knowledge and to hide useless information. For example, by aggregating data delivered in millisecond to cycles of minutes and by calculating the sum(), avg(), max(), min(), count(), and other statistical functions like the standard deviation, new information regarding the smoothness of the data delivery or their homogeneity, can be generated. In the mean time, useless information such as the initial information tracked in milliseconds cannot be any more referenced. Usually, data from multiple data sources and different domains are aggregated, for example equipment and production-line data can be combined in order to deliver useful information to engineers regarding production bottlenecks, but also reports for the upper management. Commonly, in the industry, the aggregation is performed on daily bases, using batch jobs. Usually, the batch jobs are started at night, after all data has been collected.

Cisco Systems, Inc. identified the deficiencies of the classical nightly batch jobs aggregation strategy, disclosed and summarised them within five Pain Points in the White Paper "*BI and ETL Processes Management Pain Points; Understanding the most pressing pain points and strategies for addressing them*". All Pain Points, except for Pain Point 3 regarding ad-hoc reporting, have been addressed in a conference paper [1]. Within this paper, the theoretical background presented in [1] is extended, such that it equally covers the ad-hoc reporting functionality. Ad-hoc reporting is a *Business Intelligence (BI)* process used to quickly create reports on an as-needed basis. Ad-hoc reports are generally created for one-time use to find the answer to a specific business question and offers a wealth of values to business across industries. Ad-hoc reports assures the required flexibility to be able to follow and adapt to the continually changing business environment. Furthermore, ad-hoc evaluation strategies enable the users to autonomously create new reports without involving highly qualified IT personnel.

In order to illustrate our methodology, we will present a typical adjustment regarding the statistical function *Standard Deviation (SD)*, such that it can be used within the continuous aggregation strategy. The standard deviation is simple enough to gain a good overview concerning the difficulties that arise in practice, but it is not trivial and exemplifies the immanent problem of the continuous information processing methodology. The usual representation of the standard deviation is adapted to fit our needs.

Aggregation is an operation to obtain summarised information by using aggregate functions. A new approach for information aggregation based on a very simple and straightforward starting point is formulated in this paper – namely, that within the information flow, *the process of information aggregation should be started as early as possible, best as*

soon as the collection phase is initiated. This strategy assumes a strict and clearly defined architectural design strategy of the computational framework and enables real-time capability of the system, therefore, the new methodology is termed *Continuous Information Processing Methodology* (CIPM). In order to be able to in-depth analyse the CIPM, a formal, mathematical model is set up, the conversion of the underlying structure is defined and the pros and cons of CIPM, as opposed to the classical batch jobs strategy, are discussed.

As defined in [2] "Big Data is the information asset characterised by such a *high volume*, *velocity* and *variety* to require *specific technology* and *analytical methods* for its transformation into value". According to the definition above, Big Data is much more that high volume of data and needs unconventional methods to be processed.

At the same time, a cultural change should accompany the process of investing in interdisciplinary Business Intelligence and *Data Analytics* education [2], involving the company's entire population, its members to "efficiently manage data properly and incorporate them into decision making processes" [3].

A. Motivation

1) Rapidly increasing data amount: The total amount of data created, captured, and consumed globally is forecast to increase rapidly, reaching more than 180 Zettabytes in 2025, as opposed to 64.2 Zettabytes in 2020 and 15.5 Zettabytes in 2015 [4]. Real-time information processing has become a significant requirement for the optimal functioning of the manufacturing plants [5]. Worldwide by 2022, over 50 billion *Internet of Things* (IoT) devices including sensors and actuators are predicted to be installed in machines, vehicles, buildings, and environments and/or used by humans.

2) Real-time requirements: Demand is also huge for the real-time utilisation of data streams, instead of the current batch analysis of stored Big Data [6]. The operations of a real-time system are subject to *time constraints* (deadlines), i.e., if specified timing requirements are not met, the corresponding operation is degraded and/or the quality of service may suffer and it can lead even to system failure [7]. In a real-time system deadlines must always be met, regardless of the system load. Usually, a system not specified as real-time cannot guarantee a response within any time frame. There are no general restrictions regarding the magnitude of the values of the time constraints. The time constraints do not need to be within seconds or milliseconds, as often they are understood. There is a general tendency that real-time requirements are becoming crucial requisites.

Travellers require current flight schedules on their portable devices to be able to select and book flights; in order to avoid overbooking, the flight plans and the filled seats must be kept reasonably current. Similarly, people expect instant access to their business-critical data in order to make informed decisions. Moreover, they may require up-to-date aggregated data or even ad-hoc requests. This instant access to critical information may be crucial for the competitiveness of the company [8].

B. Aim

Cisco [9] identified a couple of *Pain Points* in the Business Intelligence (BI) area, but these Pain Points carry a more general validity:

1) *the race against time*; managing batch window time constraints,
2) *cascading errors and painful recovery*; eliminating errors caused by improper job sequencing,
3) *ad hoc reporting*; managing unplanned reports in a plan-based environment,
4) *service-level consistency*; managing service-level agreements,
5) *resources*; ETL resource conflict management.

Our approach is addressing all five Pain Points.

In conclusion, continuous information processing enables a new perspective on aggregation strategies, such that aggregation is performed in parallel to the data collection phase. Preliminary aggregated values corresponding to the current state of the retrieved data are available for ad-hoc evaluation, nightly batch aggregation becomes obsolete.

C. Outline

The remainder of the paper is structured as follows: section II gives an overview regarding existing work related to the described problem. An informal presentation of the continuous aggregation strategy is presented in section III, whereby section IV introduces the mathematical model and describes the methodology to transform the batch aggregation into continuous aggregation. The presentation of the main results and discussions based upon these results constitute the content of section V, whereas section VI summarises our contributions and draws some perspectives for future work.

II. Related work

Primarily, the focus of this section is on algorithmic approaches regarding the state of the art. The analysis of different one-pass algorithms [10] and their efficient implementation is beyond the scope of this paper as well as pure technical solutions based on database technologies.

A. SB-trees

A B-tree is a balanced tree data structure, that keeps data sorted and allows searches, sequential access, and deletions in logarithmic time; the tree depth is equal at every position, whereas the SB-tree is a variant of a B-tree such that it offers high-performance sequential disk access [11], [12]. Zhang [12] outlines the key challenges of spatio-temporal aggregate computation on geo-spatial image data, focusing primarily on data having the form of raster images. Zhang gives a very detailed overview of the state of the art regarding efficient aggregate computation. Zhang's approach is based on *aggregate queries* common in the database community, including data cubes, whereas our approach (CIPM) does not focus on database

technology when calculating the aggregation functions. For example, the improved multi-version SB-tree [12] consumes more space than the size of raw data. Other approaches use only a small index, reducing the space needed, but supporting only count and sum aggregate functions. The main idea behind the SB-trees is to provide through a depth-first search, – by accumulating partial aggregate values – a fast look-up of computed values [12], [13].

B. Scotty

Scotty [14] is an efficient and general open-source operator for sliding-window aggregation for stream processing systems, such as Apache Flink, Apache Beam, Apache Samza, Apache Kafka, Apache Spark, and Apache Storm. It enables stream slicing, pre-aggregation, and aggregate sharing including out-of-order data streams and session windows [15]. The aggregate window functions are: avg(), count(), max(), min(), sum(). Being a toolkit, the out-of-the-box aggregate functions are restricted to the above. Implementation details are disclosed in a preprint paper [16]. Scotty can be extended with user-defined aggregation functions, however, these functions must be associative and invertible. Since Scotty is open source, additional user extensions are always possible.

Sliding window aggregation is also a main topic regarding this paper, even if sliding windows are not used for reporting/evaluation. There is always the possibility that erroneous data is captured and cannot be corrected automatically. This cannot be avoided, since a data set may look formally correct, but may be wrong with regard to its content. Such anomalies can be detected hours after the data has been processed and should be corrected.

According to [17] research on sliding-window aggregation has focused mainly on aggregation functions that are associative and on FIFO windows. Much less is known for other nontrivial scenarios. The question arises, whether is it possible to efficiently support associative aggregation functions on windows that are non-FIFO? Besides associativity and invertibility, what other properties can be exploited to develop general purpose algorithms for fast sliding-window aggregation? Tangwongsan et al. [18] present the Finger B-tree Aggregator (FiBA), a novel real-time sliding window aggregation algorithm that optimally handles streams of varying degrees of out-of-orderness. The basic algorithms can be implemented on any balanced tree, for example on B-trees.

C. Holistic functions

The median, which is the middle number in an ordered list of items, is a holistic function, i.e., its results have to rely on the entire input set, so that there is no constant bound on the size of the storage needed for the computation. An algorithm suitable for continuous aggregation based on heap technology can be found in [8]. For the sake of completeness, the main idea is presented hereafter. In order to store the data, two heaps are used, one for the higher part and one for the lower part of the data. The newly collected dataset is inserted into the corresponding heap; if the case arises, the heaps are balanced against each other, such that the two heaps contain the same number of items, etc. Hence, in some cases, holistic aggregation functions can be used with continuous aggregation techniques, they should however satisfy the foreseen time constraints.

Unfortunately, even such common functions as min() or max() have a holistic behaviour in some circumstances. If used for example in a sliding-window aggregation environment or if retrospectively data corrections are allowed, then all of the values have to be stored in order to determine the minimal or the maximal value of the stream. Similarly, a heap of sorted values can be used in order to implement real-time capability.

D. Quantile

A survey of approximate quantile computation on large-scale data is given by Chen [19]. In streaming models, where data elements arrive one by one, algorithms are required to answer quantile queries with only one-pass scan. Formulas for the computation of higher-order central moments or for robust, parallel computation of arbitrary order of statistical moments can be found here [20], [21], some of them are one-pass incremental approaches.

In conclusion, the main focus of the existing research has been to develop aggregate queries for efficient retrieval and visualisation of persisted data. However, with Scotty a general open-source operator for sliding-window aggregation in stream processing systems, – such as, for example, the Apache family – has been developed. Scotty incorporates the usual aggregate functions like avg(), sum(), etc., and it has the possibility to include special user defined functions. Tangwongsan [17] points out that much less is known for nontrivial scenarios, i.e., functions that are not associative and do not support FIFO windows. Our approach, however develops the strategy and technology for continuous information processing, abbreviated CIPM and shows that functions, which allow efficient one-pass implementations are suitable for CIPM. Moreover, holistic functions allowing appropriate implementation, for example median [8], can be used with CIPM.

III. Problem Description

The technical terms “information function” and “aggregation function” [22] are used synonymously within this paper. They highlight the same topic from different perspectives. Corporate reporting aims to provide all of the counterparties with the information they need in order to transact with a company. This can be termed the *information function* of corporate reporting [23]. Within this paper, we assume that the data collection and the subsequent data transformation are continuous processes, aggregation being the process that succeeds transformation. The terms “continuous information processing” and “continuous aggregation” are used alternatively, emphasising that within the continuous information processing, the continuous aggregation is the challenging part.

According to Cisco [9] “One of the biggest challenges facing an IT group is how to complete extract, transform, and load (ETL) and subsequently aggregate the traditional

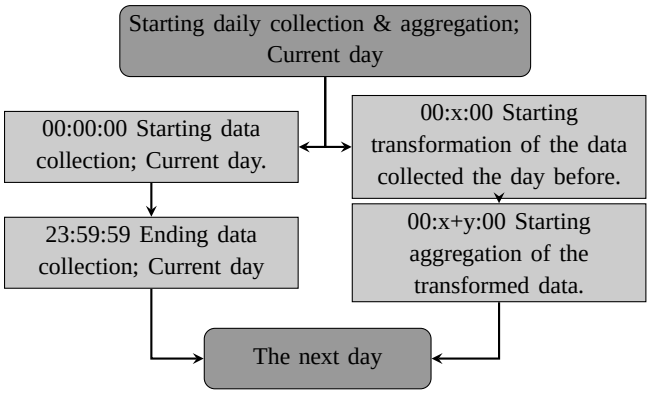


Figure 1: Simplified flow diagram exemplifying the batch job strategy(x is the time gap due to the collection delay; y is the time gap due to transformation).

batch-based Business Intelligence (BI) processes within the constraints of an ever shrinking batch window." Although there is a trend toward real-time BI, the vast majority of BI today relies heavily on batch processing [9]. Cisco identified several factors which contribute to the difficulty in managing these processes in the foreseen time frame.

Firstly, there is a severe lack of visibility into the various processes themselves, which are very complex. This means there will always be bottlenecks, and it is very difficult to predict where they will occur.

Secondly, data streams are expected to arrive in a defined time window, if they are late or corrupted, then errors may occur. Thus, in these cases, the nightly aggregation routines have to be (re)started later, including also during the usual working hours. But Cisco does not identify two major factors that impact the time frame of the batch jobs, namely the impossibility to anticipate all the patterns of data to be processed and the imperfection of the executions plans, which can lead to performance degradation. Thus, the most accurate testing covers only the patterns of data retrieved in the past or anticipated. Accordingly, even the most accurate design and testing strategy cannot guarantee the time constraints of the batch jobs.

A. Overview of the CIPM

Following, the fundamental aspects of the continuous aggregation strategy are outlined by using two simple flow diagrams, Figure 1 depicts the principles of the classical nightly jobs aggregation strategy, whereas Figure 2 describes very succinctly the continuous aggregation strategy. It is assumed within these examples, that reporting is based on daily aggregated data. Data collection starts at 00:00:00 for the current day (i.e., the considered day) and it ends, retrieving data generated till 23:59:59 of the same day. Whenever applying the classical batch jobs strategy as depicted in Figure 1, the transformation/aggregation is started only after the data is fully retrieved/collected for the considered day. Considering this use case, the transformation/aggregation for the current day can be started only on the subsequent day.

On the other hand, the CIPM (exemplified in Figure 2) is carried out on small chunks of data, for example

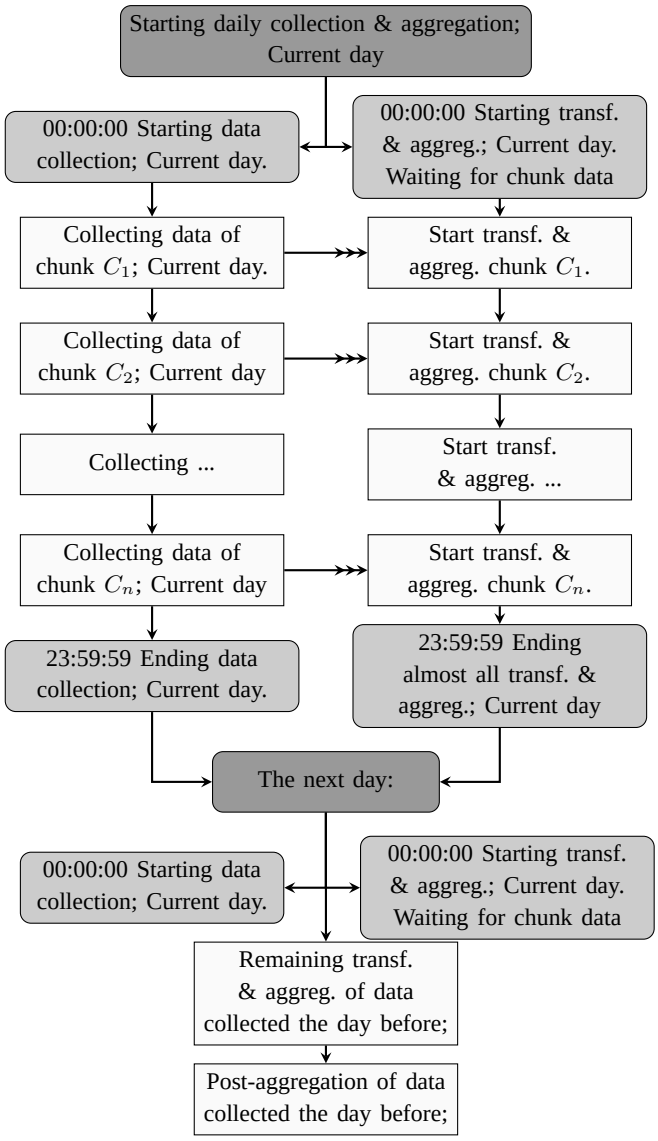


Figure 2: Simplified flow diagram exemplifying the continuous aggregation strategy. The arrow with three heads signifies that the aggregation phase waits till the respective chunk data has been collected. The post-aggregation phase of the previous day is not depicted.

$C_1, C_2, ..., C_n$, usually such that the transformation/aggregation is performed on data loaded into memory during the collection phase. This way, reloading data into memory for aggregation purposes is obsolete.

The continuous aggregation strategy is quite straightforward: after midnight, the collection phase for the current day is started, i.e., the chunks $C_1, C_2, ..., C_n$ are retrieved one after another in chronological order. While the second chunk C_2 is retrieved, transformation and aggregation are performed on the first chunk C_1, and so on and so forth. At the end of the current day, most of the collected data is aggregated. At the beginning of the subsequent day, the remaining chunk(s) are transformed/aggregated and a post-aggregation phase is started, during which the final calculations are performed. In the end, soon after midnight, the aggregated values are ready

for reporting. Similarly, after midnight, the collection/transformation is resumed for the current day.

In order to keep the presentation simple and accessible and to avoid technical complications, it is assumed that the time to perform the transformation/aggregation of a chunk is slightly lower than the corresponding time of the collection phase. In real-world systems, under some circumstances, this is obviously not necessary. Let us suppose that the time to retrieve a chunk is t, but the time to transform/aggregate the values of a chunk is slightly lower than $3t$ and let A_i be the aggregation phase of chunk C_i. Then, the start of A_i is phase-shifted by t with regard to C_i, i.e., A_1 is started simultaneously with C_2, etc. As a consequence, A_i completes before $C_{(i+4)}$ is started. Hence, in this example there are three instances of the aggregation algorithm running in parallel. Possibly, information between the aggregation instances running in parallel need to be exchanged.

Additionally, it is assumed within this paper, that the chunks are of the same size and the time to retrieve them is independent of the particular chunk. Of course, this assumption is not necessary in real-world systems.

B. Exemplification using Standard Deviation (SD)

Next, the complexity of our approach is illustrated by exemplifying the technology on the *standard deviation of the sample*. It shows how much variation (dispersion, spread) from the mean exists. The SD has been chosen for exemplification, since the adaptations in order to be used within CIPM are straightforward to be presented without being trivial.

Let $\{x_1, x_2, \ldots, x_N\}$ be the observed values of the sample items, let $\bar{x} := 1/N \sum_{i=1}^{N} x_i$ be the mean value of these observations. The common representation of the (uncorrected sample) standard deviation is:

$$SD_N := \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}. \quad (1)$$

At first glimpse, the above representation of the standard deviation cannot be applied using continuous computing techniques. The impediment is the term $\bar{x}$. In order to be able to apply the formula (1), all the data involved has to be first collected. Chan et al. [24], [25] called the above representation *two-pass algorithm*, since it requires passing through the data twice; once to compute $\bar{x}$ and again to compute SD_N. This may be unwanted if the sample is too large to be stored in memory, or when the standard deviation should be computed dynamically as the data is collected.

Regrouping the terms in the formula above, the well known representation is obtained:

$$SD_N = \frac{1}{N}\sqrt{\left|N\sum_{i=1}^{N}x_i^2 - \left(\sum_{i=1}^{N}x_i\right)^2\right|}. \quad (2)$$

The alternative representation (2) of the standard deviation is suitable to be used within the continuous computation approach.

Let $1 \leq n \leq N$, let $A_n := \sum_{i=1}^{n} x_i^2$, let $B_n := \sum_{i=1}^{n} x_i$, and let $S_n := n \cdot A_n - B_n^2$.

During the data collection phase, the values of A_n and B_n are updated, either after each item x_n is collected and known to the system, or considering small batches. Thus, at each point in time, during the data collection phase, the values of $A_{(n+1)}$ and $B_{(n+1)}$ can be easily calculated by adding up the corresponding value of the new item. Hence,

$$A_{(n+1)} = A_n + x_{(n+1)}^2.$$

Similar results hold for $B_{(n+1)}$.

Accordingly, at each point in time, the standard deviation can be easily calculated, if needed, as a function of A_n, B_n and S_n. It follows:

$$S_{(n+1)} = S_n + A_n + n \cdot x_{(n+1)}^2 - 2x_{(n+1)} \cdot B_n.$$

Hence, intermediary results and trend analysis are possible during the data collection phase .

For example, almost all *Key Performance Indicators* (KPIs) used in the semiconductor industry can be adapted to be applied within CIPM [26]–[29]. The same is true in other areas of the industry or business.

C. Reasons for choosing SD

The considerations above were drafted merely to illustrate the continuous computation technology, in real-world systems the representation (2) without using absolute values in the square root function can lead to negative values. When A_N and B_N are calculated in the straightforward way, especially when N is large and all of x-values are roughly of the same order of magnitude, rounding or truncation errors may occur [30]. Please note that the representation (2) using absolute values, has been adopted in order to avoid negative values under the square root. Using double precision arithmetic can possibly avoid the occurrence of anomalies as above.

D. Counterexample

Unfortunately, there are also some simple and well known functions, such as the *Average Absolute Deviation (AAD)*, which, generally speaking, cannot be used with continuous computing techniques; AAD is calculated as the mean of the sum of the absolute differences between a value and the central point of the group:

$$AAD_N = \frac{1}{N}\sum_{i=1}^{N}|x_i - M|.$$

The central point M can be a mean, median, mode, etc. For some distributions, including the normal distribution, AAD can be related to or approximated with the corresponding standard deviation [31]–[33].

Under some circumstances holistic functions, such as the AAD presented above, can be used within real-time applications. As already mentioned, there is no constant bound on the size for the storage needed to compute holistic functions. If the

task is to convert the nightly batch aggregation to continuous aggregation and real-time constrains are only required for the data of the last 24 hours or the current day, then obviously the size of the required storage is bounded, since the number of items expected during 24 hours can be very well estimated. Hence, if the time required to calculate the value of AAD is within the time constraints imposed, then real-time capability of the application is given. On the contrary, if real-time capability is required for example for streaming data, then this depends heavily on the size of the streamed data comprising the time interval required and the possibility to predict the size of the data. In such cases, approximations can help, see [31]–[33] for approximations based on standard deviation.

E. Flow Factor

Next, in order to illustrate the power and significance of the continuous aggregation strategy, a real-world example from the semiconductor industry [27], [29], [34] is presented. For the sake of completeness, a slightly modified part of the model is taken over from [29]; the model has been originally described for data centres. In order to be able to understand and follow the presentation, some basic data structures are introduced.

1) Description of the data structure for WIP-data: In simplified terms, the lot/wafer enters the production line and it is processed consecutively on (different) operations/equipments according to the production plans (i.e., routes) and leaves the frontend area to be systematically examined/tested in the backend area. Afterwards, the wafers are cut into segments, also called dies, which are packaged as chips and are ready to be shipped. There are two major data structures, Work in Progress (WIP)-data and Equipment-data. The WIP-data models the successive processing of the lot/wafer at different operation in the frontend area and the Equipment-data models the data associated to an equipment.

A simplified data structure for WIP-data is: (*"unit-ID", "step", "next-step", "track-out-TS", "trans-code", "equipment", "product", "unit-type", "unit-desc", "unit-value", "route"*). The unit is the manufactured item, which is tracked by the Manufacturing Execution System (MES). It can be a lot, a wafer or a die/chip. This information is tracked by the attribute "unit-desc". The attribute "unit-type" is an additional distinction between the material units, such that the units are *Productive, Development, Test, Engineering*, etc. The transaction code ("trans-code") denotes the event that is performed at a specific "step" and "equipment" during the production process. Common transaction codes in the semiconductor industry are *TrInT, TrOutT, Create a Lot, Ship a Lot*, etc. The attribute "product" characterizes the manufactured item, (like technical specifications, etc.), which can be tracked within the production process. The "step" is the finest abstraction of the processing level, which is tracked by the reporting system. It is usually the operation [8]. The attribute "next-step" denotes the succeeding "step" (i.e., operation) to which the item is transferred. Additionally, the attribute "track-out-TS" stores the time stamp of the item when it left the "step" to be processed at the "next-step". This is triggered through the transaction code TrOutT. The attribute "unit-value" contains the number of items processed at the considered "step", whereas the "route" contains the processing specification for the "product" involved.

2) Description of the aggregation functions: The following definitions are taken from [29]. Let u be a unit. We denote by $TrInT_s(u)$ the *track in time* of u, i.e., the point in time when the processing of unit u is started at "step" s. Analogously, $TrOutT_s(u)$ is the *track out time* of u, i.e., the point in time when the processing of unit u has been finished at "step" s.

The *Raw Process Time (RPT)* of a unit u related to "step" s is the minimum processing time to complete the "step" s without considering waiting times or down times. We denote the raw process time of unit u related to "step" s by $RPT_s(u)$. We assume that for a "step" s, the function $succ_s(u)$, which identifies the succeeding "step" of s for the unit u is well defined. Analogously, we assume that the history of the production process is tracked, so the predecessor function $pred_s(u)$ of each "step" s is well defined.

By *Cycle Time (CT)* we denote the time interval a unit or a group of material units spent in the system or subsystem. The cycle time of a unit $u \in U$ spent at a "step" $s \in S$ in the system can be represented as:

$$CT_s(u) := TrOutT_s(u) - TrOutT_{pred_s(u)}(u). \quad (3)$$

In order to simplify the notation to be used within the Transact-SQL syntax for SQL Server we set

$$PrevTrOutT = TrOutT_{pred_s(u)}(u).$$

Hence, using the above notation, we have the expression for the cycle-time corresponding to a unit:

$$CT = Datediff(ss, TrOutT, PrevTrOutT). \quad (4)$$

where $Datediff(ss, TrOutT, PrevTrOutT)$ returns in seconds the specified time difference.

Algorithm 1 Sample search query for the calculation of the flow factor - standard method

Input: WIP-data-table_name;
Output: Flow factor;
Result: Compound aggregated value of the flow factor ready for reporting on week 6 of 2022;

```
/* Retrieving the Flow Factor(FF)                    */
select sum(CT)/sum(RPT) as FF
from from WIP-data-table_name
where TrOutT > '06.02.2022 00:00:00'
and TrOutT <= '13.02.2022 00:00:00'
group by "step", "equipment", "product", "unit-type", "unit-desc"
```

Figure 3: SQL-code based algorithm retrieving the flow factor and exemplifying the standard method for ad-hoc reporting strategy by enabling real-time capability. For this purpose, the representation (4) of the cycle time has to be used, whereas the representation (3) is inefficient.

3) Description of the problem: The SQL-query based on the representation in Figure 3 and representation (3) is inefficient due to the fact that the chronological order of the steps is not fully specified by the "route". The human operator has the possibility to determine the subsequent "step", for example to perform some additional measurements or not. Hence, to determine the previous "step", i.e., the attribute notated as "prev-step", appropriate joints have to be set up, such that in the end the value of the attribute "PrevTrOutT" is determined. Due to the fact that processing can be done in a loop for rework processes, the item with the greatest time-stamp has to be selected. In conclusion, using suboptimal data structures for reporting can do the task of retrieving the specified data to be very difficult or impossible.

4) Solving the problem: In order to circumvent the problem as above, the data structures have to be adapted according to our guiding principle such that within the information flow, the process of information setup should be started as early as possible. Two new attributes, "prev-step" and "prev-track-out-TS" are added to the structure of WIP-data and these attributes are filled during the collection phase. The attribute "prev-step" identifies the previous step with regard to the actual "step" and "prev-track-out-TS" holds the corresponding time stamp when the item left the previous step. Since data is collected chunk-wise, the cartesian product due to the corresponding joints is relatively small and the initialisation of those attributes can be done during the collection phase.

The aggregated structure may look like this: ("step", "equipment", "product", "unit-type", "unit-desc", "cycle-time", "raw-process-time"). During the small scale aggregation phase, the values of the attributes "cycle-time" and "raw-process-time" are summed up according to the corresponding grouping. For example, if an aggregated daily table is set up, then the flow factor for the lowest granularity can be calculated by just dividing two numbers. As a consequence, the query to calculate the flow factor is reduced to a simple select on a single aggregated table, see algorithm Figure 3.

5) Example for the data structure for RTC-data : For the Equipment-data an oversimplified structure is: ("equipment-ID", "chamber-ID", "work-station", "time-stamp", "current-state"). Real Time Clock (RTC) systems record the current state of the equipment [35], hence the equipment data as above is also termed RTC-data. The current-status of the equipment can be among others [36]: *Scheduled-downtime, Unscheduled-downtime, Non-scheduled-time, Productive-time, Engineering-time*, etc. For reporting purposes, the structures of WIP-data with RTC-data are combined, for example to determine the standard deviation of the cycle time concerning an equipment or group of equipments, since a certain operation can be performed on different equipments.

In conclusion, the principles of the CIPM as opposed to the classical batch jobs strategy are presented in this section. For this purpose, two representations of the standard deviation are discussed The first representation is not suitable to be applied within CIPM, whereas on the second representation the principles of the CIPM are exemplified. For the sake of completeness, a holistic function is presented which cannot be generally applied within CIPM. Nevertheless, under some circumstances, holistic functions can be used in real-time environment, the strategy to be used depends on the use case. Lastly, a real-world problem from a semiconductor company is presented. Using some principles of CIPM such as redesigning the calculation methodology, a more energy efficient, less time consuming solution is presented, which can also be used within CIPM.

IV. The Formal Model

We further formalise the description of our methodology [1] by developing a mathematical model, in order to be able to use the advantages of the rigour of a formal approach over the inaccuracies and the incompleteness of natural languages. In order to keep our model simple, transparent, and easily comprehensible, some assumption are made, for example equal length of the chunks, etc. These assumptions are not strictly necessary in a real-world environment. Hence, within the continuous information processing strategy, ensuring the continuity of data collection phase and the continuity of the transformation phase within the CIPM are more or less straightforward. Hence, the terms continuous information processing and continuous aggregation are used synonymously within this paper.

A. General considerations

Assumption 1 (Finite streams) *We suppose that the streams are* finite, *i.e., there are two points in time, t_s, the starting and t_e, the ending point, such that within this time interval, the data is collected/transformed and aggregated.* ■

Assumption 2 (Synchronous data delivery) *We suppose that each stream delivers data at the same points in time* $\{1, 2, \ldots, T\}$. ■

Definition 1 (Large scale aggregation) *If the aggregation*

a) *occurs after the entire raw data has been previously collected,*
b) *involves technologies that process all of the collected data at once,*

then the process is termed batch aggregation mode or large scale aggregation. ■

Definition 2 (Small scale aggregation) *If:*

a) *the collected data within the interval $[t_s, t_e]$ can be split into $k \geq 2$ smaller units, also termed chunks,*
b) partial aggregation *is performed on these units, such that the final aggregation values are calculated out of the corresponding partial values of the chunks,*

then the process is termed small scale aggregation. ■

Assumption 3 (length of the chunks) *We suppose that data within the interval $[t_s, t_e]$ can be split into $k \geq 2$ disjunct equal units of length $l > 1$, i.e., $[t_s, t_e] = [t_s, t_{s+l}]$, $[t_{s+l+1}, t_{s+2l}], \ldots, [t_{s+(k-1)l+1}, t_e]$.* ■

Remark 1 *Some authors specify the terms large scale aggregation or small scale aggregation regarding their ability to perform the computation in memory. Within this paper, a more algorithmic than a technical approach is followed.* ■

The term "transformation" has been introduced for the seek of completeness. In simplified terms, it is the process of harmonisations of the data structures of the raw data from different sources, such that it is best finalised for aggregation. Moreover, during transformation, the initial streams can be combined to new ones, in order to facilitate the aggregation process. During the transformation phase, the data is also verified for accuracy. In principle, the granularity of the data is not altered during the transformation phase. In order to avoid technical complications, the concept of data transformation is not considered within the formal model, but formally, it can be considered a part of the data retrieval. Assuring continuous computation during the transformation phase is more or less straightforward.

Assumption 4 (Aggregation time) *In order to be able to continuously compute – i.e., retrieve/collect and aggregate – the time to aggregate a particular small batch does not exceed the collection time of the same batch.* ■

Remark 2 *Obviously, if Assumption (4) is not met, the aggregation cannot be performed in all circumstances during the data collection phase.* ■

Notation 1 (Streams) *Let $l_X \in \mathbb{N}$ be the number of streams and let*

$$X := \{X^{(1)}, X^{(2)}, \ldots, X^{(l_X)}\}$$

be the set of streams.

Let $\{1, 2, \ldots, T\}$ be the points in time when the data is collected and known by the system, let $1 \leq t \leq T$ and let $1 \leq l \leq l_X$. The value of the stream $X^{(l)}$ collected at time t is denoted by $x_t^{(l)}$. ■

Representation 1 *The streamed values can be represented as a matrix*

$$(x_{tl})_{1 \leq t \leq T; 1 \leq l \leq l_X} \text{ such that } x_{tl} = x_t^{(l)}.$$

■

Notation 2 (Grouping, bundle of streams) *Let $l_F \in \mathbb{N}$ be the number of the aggregation functions.*

In order to perform the computation of the streams – the aggregation functions are in general functions of several variables – a grouping

$$B := \{B^{(1)}, B^{(2)}, \ldots, B^{(l_B)}\}$$

is defined on the space of the streams, such that

$$B^{(l)} := \{X^{(l_1)}, X^{(l_2)}, \ldots, X^{(l_i)}\}.$$

and $l_F = l_B$.

Accordingly:

$$b_t^{(l)} := x_t^{(l_1)} \times x_t^{(l_2)} \times \cdots \times x_t^{(l_i)}$$

is the value of the grouping $B^{(l)}$ at time t. ■

This way, new compound streams are created.

Assumption 5 (No. of grouping = No. of functions) *In order to keep our model as simple as possible, it is supposed – without limiting the generality – that the number of groupings is equal to the number of aggregation functions.*

Notation 3 *Let*

$$F := \{F^{(1)}, F^{(2)}, \ldots, F^{(l_F)}\}$$

be the set of the aggregation functions, such that for $1 \leq l \leq l_F$

$$F^{(l)} : B^{(l)} \to \mathbb{R}.$$

■

Remark 3 (Aggregation strategy) *In order to keep the model as general as possible, small scale aggregation is considered as the overall approach.* ■

This means especially, that data is collected and aggregated in small batches.

Notation 4 (Disassembling the standard deviation) *Let $1 \leq l \leq l_F$ and let $F^{(l)} : B^{(l)} \to \mathbb{R}$ be the standard deviation, see representation (2) and let $x \in B^{(l)}$ a particular stream. Let $1 \leq t \leq T$ and let:*

$$\begin{aligned} f_t^{(l,1)}(x) &:= \sum_{i=1}^{t} x_i^2, \\ f_t^{(l,2)}(x) &:= \sum_{i=1}^{t} x_i, \\ f_t^{(l,3)}(x) &:= t \sum_{i=1}^{t} x_i^2 - \Big(\sum_{i=1}^{t} x_i\Big)^2 \\ &= t \cdot f_t^{(l,1)}(x) - (f_t^{(l,2)}(x))^2. \end{aligned} \tag{5}$$

Let $F_t^{(l)}(x)$ be the value of the function $F^{(l)}$ applied on the values subscripted by $1 \leq t \leq N$. ■

Proposition 1 (Calculation of the standard deviation) *The value $F_t^{(l)}(x)$ of the function $F^{(l)}$ can be calculated out of the values of $f_t^{(l,1)}(x), f_t^{(l,2)}(x)$, i.e., by considering $f_t^{(l,3)}(x)$, namely:*

$$F_t^{(l)}(x) = \frac{1}{t} \sqrt{\left| f_t^{(l,3)}(x) \right|}. \tag{6}$$

■

B. Information processing

In the following, it is considered that the data is retrieved and aggregated in chunks, and that before performing aggregation, the chunks are not altered.

Notation 5 (Chunk-wise processing) *Let $j, q \geq 1$, such that j is the index and q is the length of the chunks and let us suppose that the streams are retrieved in small chunks:*

$$C_j := \{C_j^{(1)}, C_j^{(2)}, \ldots, C_j^{(l_x)}\}$$

of q items, i.e., the chunk $C_j^{(l)}$ of the stream $x \in B^{(l)}$ consists of the values:

$$C_j^{(l)} := \{x^{(l)}_{((j-1)q+1)}, x^{(l)}_{((j-1)q+2)}, \ldots, x^{(l)}_{(jq)}\}.$$

■

The information is processed chunk-wise, first C_1 is retrieved. As long as the next chunk C_2 is retrieved, aggregations is performed on C_1 simultaneously, then chunk C_3 is retrieved by simultaneously aggregating chunk C_2, and so on and so forth.

During the chunk-wise processing, the values of $f^{(l,1)}_{(j+1)q}$ and $f^{(l,2)}_{(j+1)q}$ corresponding to the subsequent chunk, can be easily calculated out of $f^{(l,1)}_{jq}$ and $f^{(l,2)}_{jq}$, for example:

$$f^{(l,1)}_{(j+1)q}(x) = f^{(l,1)}_{jq}(x) + \sum_{i=1}^{q} x^2_{jq+i}.$$

The value $F^{(l)}_{jq}$ corresponding to the standard deviation at time $t = jq$ can be calculated at each "step" corresponding to the points in time $\{1, 2, ..., T\}$, or alternatively, after having reached the end of the collection phase. This phase is termed *post aggregation phase*.

Definition 3 (Post aggregation phase) *The additional calculation, which is performed after all chunks have been retrieved and aggregated, is termed post aggregation phase.*

Since the small scale aggregation should be as fast and effective as possible, the functions $f^{(l,3)}_{jq}, F^{(l)}_{jq}$ must not necessary be calculated for each chunk, as it is retrieved – if there is no requirement in this direction – they can also be calculated on a case by case basis by the tool that visualises intermediary results.

C. Truncation errors

A discussion regarding the truncation errors is beyond the scope of this paper. As already mentioned, when N is large and all of x-values are roughly of the same order of magnitude, rounding or truncation errors may occur when $f^{(l,1)}_t$ and/or $f^{(l,2)}_t$ for $1 \leq t \leq T$ are evaluated in the straightforward way [30]. A greater accuracy can be achieved by simply shifting some of the calculation to double precision, see [24], [25] for a discussion on rounding errors and the stability of presented algorithms. Barlow presents an *one-pass-through* algorithm [37], which is numerically stable and which is also suitable for parallel computing.

The scope of the presentation in this paper is merely to illustrate the technology. Of course, if a function does not allow an one-pass algorithm, it cannot be used directly for continuous computation. A classical example in this direction is the average absolute deviation, as mentioned before, in some cases there are approximate one-pass implementation of the algorithms.

D. General case

So far, the standard deviation was considered as exemplification. Now, let us formalise the continuous aggregation strategy and consider the general case:

Proposition 2 (Reassembling the partial functions)

Let $1 \leq l \leq l_F$, let $j, q \geq 1$, such that j is the index and q is the length of the chunks. Let

$$F^{(l)} : B^{(l)} \to \mathbb{R}$$

be an aggregation function such that:

a) there exists l_f real valued functions

$$f^{(l,1)}, f^{(l,2)}, \ldots, f^{(l,l_f)} \text{ defined on } B^{(l)}$$

such that for each chunk $C_j^{(l)}$, the values of

$$f^{(l,i)}_{(j+1)q} \ (1 \leq i \leq l_f)$$

can be calculated out of the values of $f^{(l,i)}_{jq}$ and $C_j^{(l)}$,

b) $F^{(l)}$ is a function of $f^{(l,i)}$ for all $1 \leq i \leq l_f$.

Then intermediary results, such as the value of $F^{(l)}_{(j+1)q}$ can be calculated out of $f^{(l,i)}_{(j+1)q}$ $(1 \leq i \leq l_f)$. ■

Assumption 6 *Let j_f be the index of the final chunk to be processed. Obviously, the composition algorithms should ensure that the value $F^{(l)}_{j_f \cdot q}$ does not depend on the size of the chunks.* ■

E. Reprocessing

In practical systems, in general, there should be a technology in place that allows recalculation of the aggregated values. This is necessary if for any reason whatsoever, some stream values are erroneous. Sometimes, it takes time to correct them, since not all wrong values can be detected and corrected automatically. Regarding the standard deviation, two new functions $df^{(l,1)}_t$ and $df^{(l,2)}_t$ can be introduced, such that

$$df^{(l,1)}_{jq}(x) := \sum_{i=1}^{q} x^2_{jq+i}$$

and

$$df^{(l,2)}_{jq}(x) := \sum_{i=1}^{q} x_{jq+i}.$$

Thus, correct and updated computed values can be achieved, for example by adding to $f^{(l,3)}_T$ the new value of $df^{(l,1)}_{jq}$ and subtracting the corresponding old value $df^{(l,1)}_{jq}$, similar considerations are valid for $df^{(l,2)}_{jq}$. This means especially, that the corresponding values for the initial chunk and the corrected chunk have to be (re)calculated. In the end, the value $F^{(l)}_{j_f \cdot q}$ has to be recalculated. As already mentioned, the above considerations are included in order to illustrate

the methodology. In practice, better suited algorithms can or should be used instead.

F. Pseudo-code algorithm exemplification

In the following, a simplified algorithm to exemplify our continuous aggregation strategy is sketched. It is based on disassembling the standard deviation $F_t^{(l)}$ using the functions $f_t^{(l,1)}, f_t^{(l,2)}, f_t^{(l,3)}$, see equation (5) and (6). In order to keep the representation of the algorithm simple, it is supposed that the chunks have the same length equal to l_{chunk}, see Assumption 3. The corresponding algorithm is presented in Figure 4. In real-world systems, the data collection may involve also time limits t_{Max}, such that these time limits restrict the length of the chunks.

1) Ad-hoc reporting including request for up-to-date data: Granularity defines the level of detail of information. The finer (higher) the level of granularity, i.e., smaller grains, the deeper the level of detail. In time-series data, for example, the granularity of the data might be at a millisecond, second, minute or hour level, the higher the level of granularity, the more information is available for analysis. The lowest level of granularity is the granularity of the raw data, i.e., of the data collected by the streams. For analysing purposes, in real-word systems, the lowest level of granularity may be to fine grained, for example, the streams deliver data within milliseconds and with the highest precision, but this low level of detail is pointless for reporting. In this case, a *pre-aggregation* is performed by calculating the average or similar of the numeric values. We suppose that the streams do not contain calculated or aggregated data. Coarser level of granularity can be achieved by aggregation.

2) Term ad-hoc: According to Merriam-Webster entry 1, the term ad-hoc means "for the particular end or case at hand without consideration of wider application". Ad-hoc reporting is a model of Business Intelligence (BI), in which reports are build and distributed by non-technical BI-users to meet individual and department information requirements. The perception in the industry is that ad-hoc reporting refers to the capability to develop reports by dragging and dropping query items from the meta-model onto a design surface [38]. This means especially that ad-hoc reporting is not an "improvised" or unplanned reporting, as the definition entry 2 of Merriam-Webster for the term ad-hoc would suggest, but a very well thought out reporting environment set up by BI experts, such that the corresponding tools can be also used by non-specialists.

3) Example from the industry: Generally speaking, in the semiconductor industry, the lowest level of granularity, which characterises the production item (wafer or integrated circuits), is the *product*. The next level of hierarchy is the *product-group*, succeeded by the *product-class*, by the *technology*, and in the end by the *production-line*. The usual/standard reporting summarises the aggregated values on product, product-group, product-class, technology, or production-line level.

Let us suppose that an equipment owner, who processes a new product on his equipment, would like to report the

Algorithm 2 Sample code exemplifying the continuous aggregation strategy

Input: Stream x
Output: $F_t^{(l)}(x)$ at each retrieval point in time t
Result: Aggregated value of the standard deviation ready for reporting at any retrieval point in time t

double precision $f^{(l,1)}(x) = 0$; // component function
double precision $f^{(l,2)}(x) = 0$; // component function
double precision $F^{(l)}(x) = 0$; // value of the standard deviation corresponding to the state of collection
int $l_{chunk} = 10,000$; // number of items of a chunk
float $[l_{chunk}]$ c ; // retrieved values of a chunk
float $[l_{chunk}]$ c_{prev} ; // data of the previous chunk
int $L_{col} = 0$; // length of the collection

// --
/* data of the length of a chunk is collected */
procedure collection() {
int $l_{cur} = 0$; // length of the collected data
repeat
  collect data into c ; // collect data into the chunk
  $l_{cur}++$; // until the chunk is full
until $l_{cur} = l_{chunk}$;
for $i = 1$ **to** l_{chunk} **by** 1 **do** $c_{prev}[i] := c[i]$; // copy c to c_{prev}
} // --

/* data of the length of a chunk is collected */
procedure aggregation() {
float$[l_{chunk}]$ x ; // contains data of the previous chunk
for $i = 1$ **to** l_{chunk} **by** 1 **do** $x[i] := c_{prev}[i]$; // copy c_{prev} to x
/* calculation of the functions composing the standard deviation */

$$f^{(l,1)}(x) := f^{(l,1)}(x) + \sum_{i=1}^{l_{chunk}} (x[i])^2;$$

$$f^{(l,2)}(x) := f^{(l,2)}(x) + \sum_{i=1}^{l_{chunk}} x[i];$$

$L_{col} := L_{col} + l_{chunk}$; // number of items collected
$F^{(l)}(x) := \frac{1}{L_{col}} \sqrt{|L_{col} \cdot f^{(l,1)}(x) - (f^{(l,2)})^2(x)|}$;// only if required
} // --

/* final calculation of the standard deviation */
procedure post-aggregation() {
$F^{(l)}(x) := \frac{1}{L_{col}} \sqrt{|L_{col} \cdot f^{(l,1)}(x) - (f^{(l,2)})^2(x)|}$;
} // --

/* start aggregation in parallel to data collection phase */
procedure void main() {
collection();
repeat
  start in parallel the threads: collection() & aggregation()
  wait until both threads have finished
until *collection phase is over* ;
aggregation(); // due to the phase shift
post-aggregation(); // final aggregated values
}

Figure 4: Pseudo-code based algorithm using standard deviation exemplifying the continuous aggregation strategy.

standard deviation for the cycle time or waiting time for the last week. The standard reporting system does not cover this case, since for example, the first three days of the week the old product is processed, then the equipment is adjusted for the new product, which is processed afterwards. The standard reporting covers the first three days of the week for the old product and the remaining days for the new product. Hence, mixed reporting, covering more products should be set up. The task is not trivial, since the standard deviation corresponding to the mixed week cannot be calculated out of corresponding values of the standard deviation of each product. The example as above illustrates also the inherent difficulties of ad-hoc reporting, it may imply substantial effort on the BI side.

G. Model extension

Next, we extend our formal model in order to cover the difficulties as above by exemplifying the methodology on the standard deviation, see algorithm Figure 5. Similar to the continuous aggregation strategy:

Proposition 3 (Calculation of compound streams)
- *let t be a point in time when streams are collected,*
- *let x and y be streams such that x contains data for the product p and y contains data for the product q,*
- *let n_t^x be the number of items of the stream x corresponding to time t,*
- *similarly, let n_t^y be the number of items of the stream y corresponding to the same point in time t,*
- *let $f_t^{(p,1)}(x)$ and let $f_t^{(q,1)}(y)$ be the corresponding values for the stream x and y respectively.*

Set

$$f_t^{((p+q),1)}(x+y) := f_t^{(p,1)}(x) + f_t^{(q,1)}(y)$$

and

$$f_t^{((p+q),2)}(x+y) := f_t^{(p,2)}(x) + f_t^{(q,2)}(y).$$

Let

$$f_t^{((p+q),3)}(x+y) := (n_t^x + n_t^y) \cdot f_t^{((p+q),1)}(x+y) - \left(f_t^{((p+q),2)}(x+y)\right)^2$$

and

$$F_t^{(p+q)}(x+y) := \frac{1}{(n_t^x + n_t^y)} \cdot \sqrt{\left| f_t^{((p+q),3)}(x+y) \right|}. \quad (7)$$

Then $F_t^{(p+q)}(x+y)$ determines the values of the standard deviation corresponding to the compound stream $x+y$ at time $t \in \{1, 2, ..., N\}$. ■

The calculation of the standard deviation relying on the formula (7) should be applied by a reporting tool within the ad-hoc reporting strategy. A closer look at the relation (7) reveals that all components like $f_t^{(p,1)}(x)$, $f_t^{(q,1)}(y)$, etc., are precalculated during the continuous aggregation period.

Remark 4 (Calculation time is stream size independent) *Hence, the calculation time for the standard deviation does not depend on the size of the stream or the point in time when the calculation is performed.*

Conclusion 1 (Real-time capability of ad-hoc reporting) *Thus, ad-hoc reporting set up as in the example above in a real-time environment has real-time capability.* ■

Algorithm 3 Sample code exemplifying the ad-hoc reporting strategy

Input: Stream x; Stream y;
Output: $F_t^{(l)}(x+y)$ at each retrieval point in time t;
Result: Compound aggregated value of the standard deviation ready for reporting at the *last* point in time $t \in \{1, 2, \ldots, T\}$ for which data has been collected;

double precision $f_t^{(l,1)}(x) = 0$; // component function
double precision $f_t^{(l,1)}(y) = 0$; // component function
double precision $f_t^{(l,2)}(x) = 0$; // component function
double precision $f_t^{(l,2)}(y) = 0$; // component function
double precision $f_t^{(l,1)}(x+y) = 0$; // calculated
double precision $f_t^{(l,2)}(x+y) = 0$; // calculated
double precision $F_t^{(l)}(x+y) = 0$; // value of the standard deviation
int $n_t(x) = 0$; // number of items belonging to Stream x collected till the point in time of the ad-hoc reporting; corresponds to the last time-stamp of the collected data
int $n_t(y) = 0$;
int $n_t(x+y) = 0$;

ad-hoc-retrieval() {
/* calculation of the functions composing the standard deviation */
$n_t(x) \Leftarrow retrieve\ the\ value\ from\ database$
$n_t(y) \Leftarrow retrieve\ the\ value\ from\ database$
$f_t^{(l,1)}(x) \Leftarrow retrieve\ the\ value\ from\ database$;
$f_t^{(l,1)}(y) \Leftarrow retrieve\ the\ value\ from\ database$;
$f_t^{(l,2)}(x) \Leftarrow retrieve\ the\ value\ from\ database$;
$f_t^{(l,2)}(y) \Leftarrow retrieve\ the\ value\ from\ database$;
$f_t^{(l,1)}(x+y) \leftarrow f_t^{(l,1)}(x) + f_t^{(l,1)}(y)$;
$f_t^{(l,2)}(x+y) \leftarrow f_t^{(l,2)}(x) + f_t^{(l,2)}(y)$;
$n_t(x+y) \leftarrow n_t(x) + n_t(y)$;
$F_t^{(l)}(x+y) := \frac{1}{n_t(x+y)} \sqrt{\left| n_t(x+y) \cdot f_t^{(l,1)}(x+y) - (f_t^{(l,2)})^2(x+y) \right|}$;
}

Figure 5: Pseudo-code based algorithm using standard deviation exemplifying the principle of the ad-hoc reporting strategy and enabling real-time capability.

H. Parallel execution of the continuous aggregation routines

So far, in order to keep the algorithms simple and transparent, simplified algorithms were presented. The aim was to exemplify the principle, such that they could be easily adapted to solve real-world situations. Next, load balancing techniques are presented. The advantage of these techniques is that it can bypass Assumption (4), which limits the aggregation time and it supposes that it does not exceed the retrieval time of the chunks.

Let j be the number of parallel instances of the aggregation routine that calculates components of the standard deviation, see procedure "aggregation" in Figure 4. This procedure has to be adapted such that it calculates partial values corresponding to the function $f^{(l,1)}$ and $f^{(l,2)}$. The implementation is quite straightforward, hence only some hints are given. Define the header as: "procedure aggregation (float[l_{chunk}] x)" Define the corresponding local variable $\delta f^{(l,1)}(x)$ and set

$$\delta f^{(l,1)}(x) := \sum_{i=1}^{l_{chunk}} (x[i])^2; \tag{8}$$

Add the statement

$$f^{(l,1)} := f^{(l,1)} + \delta f^{(l,1)}(x) \tag{9}$$

for the global variable $f^{(l,1)}$. Thus, by calling j parallel instances of the procedure "aggregation", j chunks are processed in parallel. We assumed in this example that statement (8) is the time consuming one.

I. Benefits in the software development process

1) Transparent software development: One of the outstanding advantages of the continuous aggregation strategy is the possibility to simplify and align/harmonise the set-up process of aggregation, thus leading to faster, modularised and more effective and transparent software development. This involves improved maintenance possibilities due to its conceptual unity. Moreover, people can be trained much easier on maintenance, since the software developed is not the outcome of individual abilities and unique skills, but of very well specified methodologies.

2) Paradigm shift: Lewis [39], [40] stated that *software construction is an intrinsically creative and subjective activity and as such has inherent risks*. Lewis added: *the software industry should value human experience, intuition, and wisdom rather than claiming false objectivity and promoting entirely impersonal "processes"*.

Our contribution is a "step" in setting up objective criteria regarding software developing processes, such that it *can be a science, not just an art*, paraphrasing Roetzheim's statement [41] regarding software estimate. This way, our approach facilitates the *paradigm shift* from a subjective software construction activity, towards objectively verifiable straightforward strategies. Our approach does not claim that the overall effort of the transition from large scale aggregation to small scale aggregation is diminishing, the complexity of converting multi-pass algorithms to one-pass algorithms should not be underestimated. It does require *intrinsically creative* and *subjective activity* as formulated by J.P. Lewis, but merely on the algorithmic side.

J. Real-time capability

1) Real-time systems: The term *continuous information processing* involves incessant data collection and steady aggregation, such that preliminary aggregated results corresponding to the current status of the collected data are available for evaluation purposes. Continuous processing of large amounts of data is primarily an algorithmic problem [42].

Real-time systems are subject to time constrains, i.e., their actions must be fulfilled within fixed bounds. The perception of the industry of real-time is first of all fast computation [43]. Moreover, TimeSys [44] requires the following features for a real-time system:

a) *predictably fast response to urgent events*,
b) *high degree of schedulability*: the timing requirements of the system must be satisfied at high degrees of resource usage,
c) *stability under transient overload*: when the system is over-loaded by events and it is impossible to meet all the deadlines, the deadlines of selected critical tasks must still be guaranteed.

The characterisation above exemplifies the different requirements in some fields of the industry. A real-time system requires real-time capability of the underlying components, including the operating system, etc. These considerations show the immanent difficulties of the industry to cope with the complexity of real-time requirements of opaque and incomprehensible systems.

2) Real-time capability of CIPM: In order to point out the real-time capability of a continuous information processing system, its behaviour is analysed and it is shown that it satisfies the given time limits. In real-world systems, it is supposed that the *maximum size of the streaming data* and the *streaming speed* are known and these thresholds are not exceeded.

With the aim to keep the argumentation simple and straightforward, it is assumed that the streaming speed is constant, i.e., the same amount and type of data is collected within equal time intervals. Hence, it is appropriate to setup chunks of data of the same size collected within equal time spans, such that the aggregation time of different chunks is equal. The aggregation time t_{agg} of a particular chunk should not exceed its retrieval time t_{ret}, i.e., $t_{agg} \leq t_{ret}$, else data to be aggregated will accumulate, see Assumption 4.

The strategy to achieve real-time behaviour based on continuous stream computing is straightforward.

Remark 5 (Condition for achieving real-time capability) *Let t_C be the time constraint such that within the time interval specified accordingly, aggregated data should be available. In order to have real-time capability, the condition*

$$t_{ret} + t_{agg} \leq t_C$$

should be satisfied.

Obviously, to achieve this goal, some fine tuning should be performed by choosing the appropriate size of the chunks. Hence, continuous computation including small scale aggregation, pave the way for real-time capability.

In conclusion, within this section a formal model has been introduced in order to best describe the concepts of the continuous information processing strategy. The focus is on the terms of one-pass algorithm, small scale aggregation, continuous computing, and real-time capability. One-pass algorithms

enable small scale aggregation, which can pave the way for real-time capability, on the condition that the timely constrains can be satisfied by the underlying computing environment. Actually, the one-pass requirement of the algorithms is not necessary, it suffices that the partial results of the computation of the chunks can be merged such that the expected aggregated values can be calculated.

V. Outline of the results; Discussions

Our objective has been to work towards developing practical solution to overcome the difficulties related to batch jobs, identified by Cisco in a white paper [9] as Pain Points. The pros and cons of the newly developed continuous information processing strategy versus the traditional batch jobs approach are outlined in this section and additional weak points of each technique are identified.

A. Cisco's Pain Points

1) Toughest challenge: The main challenge – which led to the outcome of this paper – was to investigate, whether it is possible to give satisfactory answers to the Pain Points raised by Cisco [9] concerning batch aggregation on data streams. Except Pain Point No. 3 regarding ad hoc reporting, to all other Pain Points, such as batch window time constraints, painful recovery, service-level agreements, etc., methods of resolution have been previously established [1]. In order to be able to properly present our methodology, a formal model has been set up and it has been shown that under some circumstances (for example, if the aggregation functions can be processed efficiently in one-step) the data collection and data aggregation can be performed continuously, and thus comprise real-time capability.

2) Sticking point – additional implementation effort: The one-pass implementation (or alternatively using small scale aggregation technology) of aggregation functions, can be meticulous, and may require additional effort. Most of the aggregation functions, also termed *measures*, used in the industry, permit such implementations; one of the well-known counterexample is the average absolute deviation. Since the computation is continuous and final results are available soon after the data collection has been completed, the Pain Point No. 1 regarding the question of batch window time constraints is obsolete.

3) Energy efficiency due to simplified recovery and to load distribution: Painful recovery (Paint Point No. 2) is less painful if there is a well thought-through recovery algorithm in place, such that only the erroneous parts are recalculated. Since there is a much better control of the computation/aggregation flow, a better service-level and resource conflict management can be achieved by using the continuous aggregation strategy.

It is true, that usually, batch jobs are performed during nighttime hours, when the workload on the computer is lower than during working hours. Unfortunately, due to computation errors or erroneous raw data, the batch aggregation has to be restarted also during normal working hours. Hence, the computer capacity should support the extended load due to recomputing the batch jobs during working hours. On the contrary, by using continuous computation, the load is distributed uniformly over the whole duration of the data collection and as a result, peak loads remain manageable. Moreover, due to our aggregation strategy – such that calculation is performed during the collection phase as early as possible, best when the data is still in memory – reloading the persisted data into memory is reduced to a minimum. Besides, the small scale aggregation can be optimised by identifying the optimal size of the chunks, such that the time constraints are met with minimal computational effort. This way, smaller computers can be used, especially since the energy efficiency of the batch aggregation is in general significantly worse than the correspondent computation regarding small scale aggregation.

B. Continuous aggregation versus batch jobs

1) Our fundamental computational strategy in a nutshell: According to the long time experience of the first author, the best performance in the field of Business Intelligence/Data Warehousing is obtained if the data is processed/transformed/precalculated as soon as possible; best, *as soon as the data is known to the system*. This includes also multiple storage strategies of the same raw/transformed data. Sometimes, it is advantageous to pursuit a *dual strategy*. On the one hand, try to follow the continuous computation strategy as long as possible i.e., as long as the implementation of the corresponding aggregation functions is possible with reasonable effort and run-time performance, and on the other hand, precalculate as much as possible by maintaining the batch jobs strategy.

2) Executions plans as the weak point of the batch jobs strategy: The main challenge of the batch jobs strategy, when using general purpose database management systems, is a technical one and it relates to the optimisation through *execution plans*. In highly simplified terms, the execution plans attempt to establish the most efficient execution of statements (queries) out of a summary of pre-calculated statistics. Unfortunately, the execution plans do not always generate the optimal (fastest, most efficient) query; performance can also degrade if the execution plans are updated. Hence, if the streams are not steady, performance degradation of the batch jobs may occur. There are methods to overcome the automatic generation of the execution plans, but the problem in principle remains.

On the contrary, by using small scale aggregation, the size of data sets on which computation is performed is more or less constant, and data is in memory, hence less prone to fluctuations due to the executions plans. It is therefore reasonable to assume some upper bounds, enabling real-time capability of the system.

C. Enhanced system modelling

One of the most important side benefits of the continuous information processing strategy is the straightforward system modelling. In this way, the design of the architecture, data flow, aggregation strategy, database schema design, etc., is given by the structure of the streaming data, the aggregation

functions and the algorithms of their implementation. Thus, the more individualistic design, heavily based on the experience of the application developer is converted into a predefined set of well founded modelling strategies, sustaining a paradigm switch from more or less subjectively individualistic conceptions in software design and development towards objectively established optimal solutions. Quantitative estimations show that many Data Warehouse projects fail at a concerning rate, wasting all the time, money, and effort spent on them [45].

D. Broadening the tasks of the classical reporting strategy

1) Supporting production control: The essence of the continuous information processing strategy is that it enables the calculation of the aggregation functions during the collection phase. For example, for reporting purposes, the data for a full day is collected. The classical batch jobs strategy envisaged the generation of the data pool for reporting only after the data has been fully collected. Hence, calculated/aggregated values for reporting were available on the next day, depending on the execution time of the batch jobs. Thus, the scope of classical reporting strategy was to capture, survey and review the production status of the previous day. On the contrary, based on the data already collected, the continuous aggregation strategy enables the calculation/generation of preliminary reports at various points in time. This way, for example, soon after 12:00, the daily reports show the production figures corresponding to the time frame [0:00, 12:00]. Therefore, if these figures are not optimal, corresponding measures to boost production could be taken. Thus, modern reporting based on our technology enables *production control*.

2) Reducing ramp-up time: In some cases, optimisation can be substantial, saving time and costs. For example, in the semiconductor manufacturing, there are optional production steps, where the material is measured. The number of measurements can be in the range of hundreds, and the measurement time can last for several hours. The common aggregation technology assumes that all measurement data is collected before starting the aggregation. By adopting our continuous computation technology, preliminary measurement results can be calculated. This way, faulty processed material can be identified earlier, since there is no need for full calculation in order to identify faulty processing, and hence, the ramp-up time of a new product can be substantially reduced, thus giving the company decisive advantage over his competitors.

E. Additional issues

1) Hardware upgrade vs. performance improvement: Next, two issues are addressed, which are decisive from technical point of view:

1) absence of optimal Data Warehouse design methodology,
2) performance problems due to the high complexity, requirements on expandability, and the low scalability of the existing complex solutions.

According to the experience of the first author at Qimonda in the Business Intelligence and Data Warehouse environment, increasing the processing capability of the computers does not always lead to improved performance of the Data Warehouse applications. By doubling the computing capacity, roughly 20% in performance improvement has been achieved. Using high performance racks produced the best results. In the end, when the effort for performance improvement is greater than the effort to redesign the Data Warehouse, appropriate measures should be taken.

2) Improving data quality: Furthermore, due to our modular straightforward design strategy, the flow of data can be much closely monitored, hence superior *data quality* can be achieved. Moreover, enhanced *maintenance* and straightforward *implementation* can be reached due to the harmonised approach.

In conclusion, the price for achieving continuous aggregation may be high, the build in functions like standard deviation cannot be used any more, and as the case may be, new one-pass or similar algorithms for the aggregation functions have to be set up, hence algorithmic and programming effort may increase. The benefits are obvious, a straightforward design strategy, up-to-date aggregated values during data collection, a uniform computational effort over the data collection period and an efficient recalculation strategy, which lead in the end to a much efficient utilisation of computational resources. Improving the performance of batch jobs is tedious, if the redesign strategy is not an option, sophisticated data base technologies or costly high performance racks can overcome the problem.

VI. Conclusion and Future Work

In the following, the advantages of the CIPM are summarised and the future work, we are concerned with, is sketched.

A. Conclusion

Satisfactory solutions to the problems caused by the nightly batch aggregation – as pointed out by Cisco [9] – are given. To ensure an accurate presentation of our methodology, a formal model has been set up and it has been shown that for a specific type of aggregation functions – including those that support efficient one-pass implementation – the data aggregation can be performed continuously and thus allows real-time capability.

1) Advantages CIPM - Résumé: Continuous aggregation strategy:

1) supports *real-time capability*; if time constraints can be met,
2) supports *aggregated values corresponding to the captured data*; i.e., reporting capability at any point in time during data collection,
3) supports *enhanced production control* due to up-to-date aggregated data at any point in time during data collection,
4) supports *straightforward design strategies* due to clear, easy understandable architectural and implementation principles,

5) supports *easy maintenance* due to transparent and straightforward software development process,
6) supports *higher quality of aggregated data* due to the simplified architectural and implementation principles,
7) supports *uniform load of the underlying database system* due to the continuous aggregation principles,
8) supports *ad-hoc real-time reporting capability* due to the principles of the continuous aggregation as elaborated by us,
9) supports *higher degree of reporting flexibility than the MOLAP technology* due to the fact that the action of slicing and dicing is not bound to a hierarchical tree model,
10) avoids *complicated SQL-queries for data retrieval or data aggregation using large cartesian products* due to the new architectural principles,
11) supports more accurate *"Knowledge Discovery in Databases" capability* due to the much detailed data analysis capacity,
12) supports *early detection of erroneous data sets* due to the continuous aggregation principles, avoiding panic situations as in the case of nightly batch jobs,
13) supports *straightforward performance improvement strategies* due to the simplified architectural principles,
14) *avoids or reduces "hot working phases" at night for the IT personnel* due to the absence of nightly jobs,
15) reduces *the risk of incomplete or missing standard reporting* – "race against time", as termed by Cisco – due to the enlarged aggregation window,
16) reduces *the complexity of the database administration* due to the new streamlined environment versus a database pushed to its limits, as in case of batch jobs,
17) supports improved *load balancing* due the modularised aggregation strategy,
18) supports *lower memory requirements*, since the data to be aggregated is already uploaded into memory during the data collection phase,
19) supports *streamlined SQL-statements*, since the data to be aggregated is preserved in small chunks,
20) avoids *regular performance improvement tasks* due to the streamlined architecture of the aggregation strategy,
21) avoids *performance bottlenecks* due to the recalculation of the nightly aggregation during business hours,
22) ensures *very good scalability, both for the small scale aggregation and for the ad-hoc reporting* due to much better performance control,
23) avoids *performance fluctuation* due to imperfect execution plans,
24) supports *efficient recalculation of aggregated values*, in case erroneous data is collected,
25) supports *parallelisation* beyond the built-in facilities of the underlying database system,
26) supports *independency of the underlying database system*, such as Relational, No-SQL, etc., and last but not least:
27) ensures *energy efficiency*, since smaller hardware can be used due to the fact that aggregation is performed during the whole data collection period.

2) Difficulties CIPM - Résumé:

1) some *predefined formulas, e.g., STDEV, cannot be used directly*, since the function above is not cumulative,
2) *difficult architectural set-up*, i.e., new algorithms have to be designed and implemented,
3) *longer development times* due to the new architectural design strategy,
4) *IT staff has to be additionally trained* due to unconventional architectural and maintenance strategies,
5) *heterogeneous team including mathematicians and data scientist* need to be used, i.e., the algorithmic part of the development may be sophisticated,
6) *increased development costs* due to the unconventional development strategies, and last but not least:
7) *strong management commitment to overcome the difficulties* due to the anticipating challenges.

We have not experienced major difficulties in implementing the CIPM approach in database applications, implementation from the scratch is pretty straightforward, porting to CIPM an existing legacy database application using batch jobs, can be quite cumbersome. For sophisticated legacy applications, the most efficient method is to try to improve performance as long as possible by applying database technologies, and/or by using high performance racks, etc.

3) Final considerations: The price of the advantages as above depends on the structure of the aggregation function. Most of the key performance indicators used in the industry permit an incremental representation, i.e., as functions of different representation of sum(), avg(), count(), similar to the standard deviation, as presented in this article. The effort in this cases is manageable. Generally speaking, the aggregation functions should permit efficient one-pass calculation, or in the case of holistic functions, an algorithm, such that the time constraints can be satisfied.

In conclusion, for small applications, where real-time constraints are not an issue, batch jobs (large scale aggregation) will deliver satisfactory results, whereas for large applications, even if real-time capabilities are not required, the advantages of CIPM may prevail.

B. Future Work

The Pain Point No. 3 of Cisco's white paper [9]: "ad hoc reporting; managing unplanned reports in a plan-based environment", has been handled in this paper. The question still remains, as to what extent "unforeseen" reports can be meaningfully set up. Furthermore, one asks oneself, what is the optimal strategy regarding volatile versus persistent aggregation, i.e., aggregation within a query set up by a visualisation tool versus aggregation persisted in a data storage. From an algorithmic perspective, persistent aggregation offers more advantages if the query is often invoked. Moreover, the results can be much better validated if the data is persisted. On the other hand, sporadic queries should remain volatile, i.e., the result of the queries should not be persisted for further reuse.

Tangwongsan [17] points out that “much less is known for nontrivial scenarios”, i.e., “functions that are not associative and do not support FIFO windows”. Within this paper satisfactory answers are given for those aggregation functions that can be reduced to additive functions, and/or allow one-pass algorithm. For example, it has been shown that the median [8] can successfully be used within CIPM.

Generally speaking, the integration of the holistic functions within CIPM remains an outstanding challenge. Commonly, for data warehouse applications, the raw data is always stored for a certain period of time in order to be able to retrace the computation. Hence, improving the speed of the computation by using distributed algorithms, etc., can ensure real-time capabilities. On the contrary, when dealing with long time and high amount of streaming data, storing the data even temporarily is not possible. In such cases, developing suitable algorithms for approximate calculations could help. Each holistic function should be handled on a case-by-case basis. A general strategy seems improbable for the time being. In conclusion, in general, for non-holistic functions (including the usual KPIs used in business and industry) satisfactory results can be given. The problem regarding the holistic functions is still open, some of them can be used under some circumstances in real-time environment.

Acknowledgment

We appreciate the information and insightful comments provided by the anonymous reviewers, which significantly improved the quality of this paper. Last, but not least, we acknowledge the assistance and helpful comments provided by Prof. Dini.

References

[1] M. Zinner, K. Feldhoff, and W. E. Nagel, “Continuous Information Processing Enabling Real-Time Capabilities: An Energy Efficient Big Data Approach,” *ICSEA 2021 : The Sixteenth International Conference on Software Engineering Advances*, pp. 155–165, 2021, Retrieved: June 2022. [Online]. Available: https://www.thinkmind.org/articles/icsea_2021_2_180_10095.pdf

[2] A. De Mauro, M. Greco, and M. Grimaldi, “A formal definition of big data based on its essential features,” *Library Review*, 2016, Retrieved: June 2022. [Online]. Available: https://www.researchgate.net/publication/299379163_A_formal_definition_of_Big_Data_based_on_its_essential_features

[3] H. U. Buhl, M. Röglinger, F. Moser, and J. Heidemann, “Big Data,” *Business & Information Systems Engineering*, vol. 5, no. 2, pp. 65–69, 2013, Retrieved: June 2022. [Online]. Available: https://doi.org/10.1007/s12599-013-0249-5

[4] Statista, “Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025,” 2021, Retrieved: June 2022. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[5] R. Sousa, R. Miranda, A. Moreira, C. Alves, N. Lori, and J. Machado, “Software tools for conducting real-time information processing and visualization in industry: An up-to-date review,” *Applied Sciences*, vol. 11, no. 11, p. 4800, 2021, Retrieved: June 2022. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/4800

[6] K. Yasumoto, H. Yamaguchi, and H. Shigeno, “Survey of real-time processing technologies of iot data streams,” *Journal of Information Processing*, vol. 24, no. 2, pp. 195–202, 2016, Retrieved: June 2022. [Online]. Available: https://doi.org/10.2197/ipsjjip.24.195

[7] I. Sommerville, “Software engineering 9th edition,” *ISBN-10*, vol. 137035152, p. 18, 2011.

[8] Zinner *et al.*, “Real-time information systems and methodology based on continuous homomorphic processing in linear information spaces,” 2015, Retrieved: June 2022. [Online]. Available: https://patentimages.storage.googleapis.com/ed/fa/37/6069417bdcc3eb/US20170032016A1.pdf

[9] Cisco, “BI and ETL Process Management Pain Points,” *White Paper*, pp. 1–9, 2010, Retrieved: June 2022. [Online]. Available: http://download.101com.com/tdwi/ww29/cisco_bi_etl_process_management_pain_points.pdf

[10] N. Schweikardt, “One-pass algorithm.” 2009, Retrieved: June 2022. [Online]. Available: http://www.tks.informatik.uni-frankfurt.de/schweika/downloads/EncycDBS_OnePassAlgos.pdf

[11] P. E. O’Neil, “The sb-tree an index-sequential structure for high-performance sequential access,” *Acta Informatica*, vol. 29, no. 3, pp. 241–265, 1992, Retrieved: June 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.9482&rep=rep1&type=pdf

[12] J. Zhang, “Spatio-temporal aggregation over streaming geospatial data,” in *Proceedings of the 10th International Conference on Extending Database Technology Ph. D. Workshop*. Citeseer, 2006, Retrieved: June 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.8746&rep=rep1&type=pdf

[13] J. Yang and J. Widom, “Incremental computation and maintenance of temporal aggregates,” *The VLDB Journal*, vol. 12, no. 3, pp. 262–283, 2003, Retrieved: June 2022. [Online]. Available: http://ilpubs.stanford.edu:8090/482/1/2000-6.pdf

[14] TU-Berlin-DIMA, “Scotty: Efficient window aggregation for out-of-order stream processing,” *Generated by GitHub Pages*, 2021, Retrieved: June 2022. [Online]. Available: https:21:3103.09.202121:3103.09.202121:3103.09.202121:3103.09.202121:3103.09.2021//tu-berlin-dima.github.io/scotty-window-processor/

[15] J. Traub, P. M. Grulich, A. R. Cuellar, S. Breß, A. Katsifodimos, T. Rabl, and V. Markl, “Scotty: Efficient window aggregation for out-of-order stream processing,” in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1300–1303, Retrieved: June 2022. [Online]. Available: https://hpi.de/fileadmin/user_upload/fachgebiete/rabl/publications/2018/ScottyICDE2018.pdf

[16] J. Traub, P. M. Grulich, A. R. Cuéllar, S. Breß, A. Katsifodimos, T. Rabl, and V. Markl, “Scotty: General and efficient open-source window aggregation for stream processing systems,” *ACM Transactions on Database Systems (TODS)*, vol. 46, no. 1, pp. 1–46, 2021, Retrieved: June 2022. [Online]. Available: https://www.redaktion.tu-berlin.de/fileadmin/fg131/Publikation/Papers/Traub_TODS-21-Scotty_preprint.pdf

[17] K. Tangwongsan, M. Hirzel, and S. Schneider, “Sliding-window aggregation algorithms.” 2019, Retrieved: June 2022. [Online]. Available: http://hirzels.com/martin/papers/encyc18-sliding-window.pdf

[18] ——, “Optimal and general out-of-order sliding-window aggregation,” *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1167–1180, 2019, Retrieved: June 2022. [Online]. Available: https://www.scott-a-s.com/files/vldb2019_fiba.pdf

[19] Z. Chen and A. Zhang, “A survey of approximate quantile computation on large-scale data,” *IEEE Access*, vol. 8, pp. 34 585–34 597, 2020, Retrieved: June 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9001104

[20] P. P. Pebay, T. Terriberry, H. Kolla, and J. C. Bennett, “Formulas for robust, parallel computation of arbitrary-order, arbitrary-variate, statistical moments with arbitrary weights and compounding.” Sandia National Lab.(SNL-CA), Livermore, CA (United States); The Xiph. Org , Tech. Rep., 2015, Retrieved: June 2022. [Online]. Available: https://www.osti.gov/servlets/purl/1504207

[21] P. Pébay, T. B. Terriberry, H. Kolla, and J. Bennett, “Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights,” *Computational Statistics*, vol. 31, no. 4, pp. 1305–1325, 2016, Retrieved: June 2022. [Online]. Available: https://www.osti.gov/servlets/purl/1426900

[22] C. Labreuche, “A formal justification of a simple aggregation function based on criteria and rank weights,” in *Proc. DA2PL2018, From Multiple Criteria Decis. Aid Preference Learn.*, 2018, pp. 1–1, Retrieved: June 2022. [Online]. Available: http://da2pl.cs.put.poznan.pl/programme/detailed-programme/da2pl2018-abstract-14.pdf

[23] R. Eccles and G. Serafeim, “Corporate and integrated reporting: A functional perspective,[w:] corporate stewardship: Achieving sustainable effectiveness, red,” *E. Lawler, S. Mohrman, J. OToole, Greenleaf*, Posted:

2 Feb 2014 Last revised: 24 May 2018, Retrieved: June 2022. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2388716

[24] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Algorithms for computing the sample variance: Analysis and recommendations," *The American Statistician*, vol. 37, no. 3, pp. 242–247, 1983, Retrieved: June 2022. [Online]. Available: http://www.cs.yale.edu/publications/techreports/tr222.pdf

[25] ——, "Updating formulae and a pairwise algorithm for computing sample variances," in *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, 1982, pp. 30–41, Retrieved: June 2022. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA083170.pdfP

[26] W. Hopp and M. Spearman, *Factory Physics: Third Edition*. Waveland Press, 2011.

[27] W. Hansch and T. Kubot, "Factory Dynamics Chapter 7 Lectures at the Universitaet der Bundeswehr Muenich," p. 68, Retrieved: June 2022. [Online]. Available: https://fac.ksu.edu.sa/sites/default/files/Factory%20Dynamics.pdf

[28] C.-F. Lindberg, S. Tan, J. Yan, and F. Starfelt, "Key performance indicators improve industrial performance," *Energy procedia*, vol. 75, pp. 1785–1790, 2015, Retrieved: June 2022. [Online]. Available: https://doi.org/10.1016/j.egypro.2015.07.474

[29] M. Zinner *et al.*, "Techniques and Methodologies for Measuring and Increasing the Quality of Services: a Case Study Based on Data Centers," *International Journal On Advances in Intelligent Systems, volume 13, numbers 1 and 2, 2020*, vol. 13, no. 1 & 2, pp. 19–35, 2020, Retrieved: June 2022. [Online]. Available: http://www.thinkmind.org/articles/intsys_v13_n12_2020_2.pdf

[30] W. Kahan, "Pracniques: further remarks on reducing truncation errors," *Communications of the ACM*, vol. 8, no. 1, p. 40, 1965.

[31] T. Pham-Gia and T. Hung, "The mean and median absolute deviations," *Mathematical and Computer Modelling*, vol. 34, no. 7-8, pp. 921–936, 2001, Retrieved: June 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0895717701001091

[32] R. C. Geary, "The ratio of the mean deviation to the standard deviation as a test of normality," *Biometrika*, vol. 27, no. 3/4, pp. 310–332, 1935.

[33] J. K. Patel and C. B. Read, *Handbook of the normal distribution*. CRC Press, 1996, vol. 150.

[34] G. Laipple, S. Dauzère-Pérès, T. Ponsignon, and P. Vialletelle, "Generic data model for semiconductor manufacturing supply chains," in *2018 Winter Simulation Conference (WSC)*. IEEE, 2018, pp. 3615–3626, retrieved: June 2022. [Online]. Available: http://simulation.su/uploads/files/default/2018-laipple-dauzere-peres-ponsignon-vialletelle.pdf

[35] F. Biebl, R. Glawar, A. Jalali, F. Ansari, B. Haslhofer, P. de Boer, and W. Sihn, "A conceptual model to enable prescriptive maintenance for etching equipment in semiconductor manufacturing," *Procedia CIRP*, vol. 88, pp. 64–69, 2020, retrieved: June 2022. [Online]. Available: https://doi.org/10.1016/j.procir.2020.05.012

[36] K. Hilsenbeck, "Optimierungsmodelle in der Halbleiterproduktionstechnik," Ph.D. dissertation, Technische Universität München, 2005, retrieved: June 2022. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss20050808-1721087898

[37] J. L. Barlow, "Error analysis of a pairwise summation algorithm to compute the sample variance," *Numerische Mathematik*, vol. 58, no. 1, pp. 583–590, 1990, Retrieved: June 2022. [Online]. Available: https://de.booksc.eu/book/6543977/98912d

[38] GSA, "Cognos Ad-Hoc Reporting (Basics)," *ePM Quick Reference Guide 75*, 2016, Retrieved: June 2022. [Online]. Available: https://www.gsa.gov/cdnstatic/QRG.075_Ad_Hoc_Reporting_6.0.pdf

[39] J. Lewis and T. Disney, "Large limits to software estimation," *ACM Software Engineering Notes*, vol. 26, no. 4, pp. 54–59, 2001, Retrieved: June 2022. [Online]. Available: http://scribblethink.org/Work/Softestim/kcsest.pdf

[40] J. Lewis, "Mathematical limits to software estimation: Supplementary material," *Stanford University*, 2001, Retrieved: June 2022. [Online]. Available: http://scribblethink.org/Work/Softestim/softestim.html

[41] W. H. Roetzheim and R. A. Beasley, *Software project cost schedule estimating: best practices*. Prentice-Hall, Inc., 1998.

[42] B. Evgeniy, "Supercomputer beg with artificial intelligence of optimal resource use and management by continuous processing of large programs," *Glob Acad J Econ Buss*, vol. 1, pp. 21–26, 2019, Retrieved: June 2022. [Online]. Available: https://gajrc.com/media/articles/GAJEB_11_21-26_zOIbTWD.pdf

[43] E. A. Lee, "What is real time computing? a personal view." *IEEE Des. Test*, vol. 35, no. 2, pp. 64–72, 2018, Retrieved: June 2022. [Online]. Available: https://ptolemy.berkeley.edu/projects/chess/pubs/1192/Lee_WhatIsRealTime_Accepted.pdf

[44] TimeSys Corporation, "The concise handbook of real-time systems," *TimeSys Corporation Pittsburgh, PA, Version 1.3*, pp. 1–65, 2002, Retrieved: June 2022. [Online]. Available: https://course.ece.cmu.edu/~ece749/docs/RTSHandbook.pdf

[45] D. Asrani, R. Jain, and U. Saxena, "Data Warehouse Development Standardization Framework (DWDSF): A Way to Handle Data Warehouse Failure," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 19, pp. 29–38, 2017, Retrieved: June 2022. [Online]. Available: http://www.iosrjournals.org/iosr-jce/papers/Vol19-issue1/Version-2/E1901022938.pdf

An Approach for Learning Behavioural Models of Communicating Systems

Sébastien Salva
LIMOS - UMR CNRS 6158
University Clermont Auvergne, France
email: sebastien.salva@uca.fr

***Abstract*—This paper is concerned with recovering formal models from event logs collected from communicating systems. We refer here to systems made up of components interacting with each other by data networks and whose communications can be monitored, e.g., Internet of Things (IoT) systems, distributed applications or Web service compositions. Our approach, which we call CkTailv2, aims at generating, from en event log, one Input Output Labelled Transition System (IOLTS) for every component participating in the communications and one graph illustrating the directional dependencies with the other components. These models can help engineers better and quicker understand how a communicating system behaves and is structured. They can also be used for bug detection or for test generation. Compared to other model learning approaches specialised for communicating systems, CkTailv2 improves the precision of the generated models by integrating algorithms that better recognise sessions in event logs. CkTailv2 revisits and extends a first approach by simplifying the set of requirements and assumptions in order to increase its applicability on communicating systems. It now integrates two new trace extraction algorithms: the former segments event logs into traces by trying to detect sessions; the latter assumes event logs to include session identifiers and allows to quicker generate models. We report experimental results obtained from 10 case studies and show that CkTailv2 has the capability of producing precise models in reasonable time delays.**

Index Terms*—*Reverse engineering; Model learning; Event Log; Communicating systems.

I. Introduction

Model learning is a software reverse engineering approach, which is receiving growing attention as a solution to help device models as state machines. These models, which capture system behaviours, can be considered as documentation or exploited in some software engineering stages, e.g., robustness or security testing. Over the last decade, there has been an extensive body of work in this field, making emerge two main categories of approaches called active and passive model learning. Such approaches infer behavioural models of systems seen as a black-boxes, either by analysing a set of execution traces resulting from monitoring (passive approaches, e.g., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) or by interacting with them (active approaches, e.g., [11, 12, 13, 14, 15, 16, 17]).

We however observed that a few works [9, 18, 19] focused on the learning of models for communicating systems. Yet, these systems are more and more omnipresent in our daily life, especially with the emergence of Internet of Things (IoT) systems. Model learning would greatly ease the understanding and analysis of communicating systems. For instance, the generation of models expressing the behaviours of every component could help engineers to quicker understand the functioning of the whole system and would assist them in the bug or vulnerability detection. We also noticed that several issues remain open in the previous approaches. For instance, the active technique given in [18] requires to know the system topology in advance and only supports accessible and testable components to build models. But, we have often observed that many communicating systems integrate untestable components. For instance, an autonomous component that continuously delivers messages is uncontrollable and hence cannot be experimented to get observations. The two other papers propose passive approaches, which do not rely on these requirements. Instead, they analyse execution traces to recover behaviours. In order to build precise models, one key point is to be able to recognise sessions in event logs, i.e., a temporary message interchange among components forming a behaviour of the whole system from one of its initial states to one of its final states. Unfortunately, these approaches cannot extract sessions. These observations motivated us to present a first approach and tool called Communicating system k-Tail, shortened CkTailv1 [2]. To design it, we choose to extend the k-Tail learning algorithm [3] with the capability to build one model called Input Output Labelled Transition System (IOLTS) for every component of a communicating system under learning. k-Tail is well-known to quickly build generalised models from traces, but it is unable to take into account the notion of component and to construct models from event logs. We showed that CkTailv1 builds more precise models than the two previous passive approaches, but we also concluded that its requirements and assumptions are still too restrictive to be practical.

In [1], we also proposed a passive model learning approach for recovering models as IOLTSs from event logs. We however assumed that correlation mechanisms, e.g., execution trace identifiers, are employed to propagate context IDs in event logs. The major contribution of this approach is its capability to automatically retrieve conversations from event logs, without having any knowledge about the used correlation mechanisms. Our algorithm is based upon a formalisation of the notion of correlation patterns and is guided towards the most relevant conversation sets by evaluating conversation quality.

Contributions: this paper presents an extension of [1] and [2], simply called CkTailv2, and the related tool. This new approach

aims at relaxing some requirements of CkTailv1 for targeting more communicating systems. CkTailv2 indeed accepts event logs having communication and non-communication events, the latter being often used to keep track of debug outputs or errors. Event logs can now integrate requests followed by an unlimited amount of responses. Besides, CkTailv2 relies on two session extraction algorithms. The former segments event logs by trying to detect sessions with respect to constraints related to the request-response pattern, the recognition of nested requests, time delays and data dependency among components. The latter assumes event logs to include session identifiers and uses the trace extraction algorithm presented in [1]. CkTailv2 also infers dependency graphs, which show in a simple way the directional dependencies observed among components. We believe that this kind of graph completes the behavioural models and will be helpful to evaluate different kinds of model properties, e.g., testability or security.

This paper also provides a detailed empirical evaluation, which investigates the precision of the models derived by CkTailv2 and its performance in terms of execution times. This empirical evaluation was carried out on event logs collected from 10 case studies and compares our implementation of CkTailv2 against three other tools namely CSight, the algorithm given in [19] and CkTailv1. This evaluation shows that CkTailv2 infers more precise models than the three previously approaches, in reasonable time delays.

In summary, the major contributions of this paper are:

- the presentation of the CkTailv2 tool and approach, which generates behavioural models and dependency graphs for every component of a communicating system from event logs,
- the design of two new algorithms allowing to better recognise sessions in event logs, and hence to build more precise models,
- the implementation of the approach publicly available in [20] and an evaluation that compares CkTailv2 with CSight, the approach proposed in [19] and CkTailv1.

Paper organisation: Section II discusses related work and presents our motivations. We provide an overview of our tool along with its capability of inferring models of communicating systems with a concrete example of IoT system in Section III. Our algorithms are detailed in Section IV. We recall some basic definitions about the IOLTS model and we describe the four steps of the approach. Section V examines experimental results and discusses about the threats to validity. Section VI summarises our contributions and draws some perspectives for future work.

II. Related Work

Model learning can be defined as *a set of methods that recover a specification by gathering and analysing system executions and concisely summarising the frequent interaction patterns as state machines that capture the system behaviour* [21]. Model learning algorithms can be organised into two main categories: active and passive approaches. Both categories are discussed below.

A. Active Model Learning

In this first category, systems are repeatedly queried (often with tests) to collect positive or negative observations, which are analysed and generalised to produce models [11, 12, 13, 14, 15, 16, 17]. Most of the active techniques have been conceived upon two concepts, the $\mathcal{L}^*$ algorithm [11] and incremental learning [12]. This model learning category is actively studied to make the approaches more effective and efficient. Among the possible research directions, some works recently proposed optimisations to reduce the query number [22], while others tackled systems having specific constraints [17].

Some active model learning approaches have been proposed for communicating systems. Groz et al. introduced an algorithm to generate a controllable approximation of components through active testing [23]. This kind of active technique implies that the system is testable and can be queried. The learning of the components is done in isolation. A recent work lifts this constraint by testing a system with unknown components by means of a SAT solving method [18]. Tappler et al. also proposed a model-based testing technique for IoT systems [24]. This technique is based on the generation of models from multiple implementations of a common specification, which are later pair-wise cross-checked for equivalence. Any counterexample to equivalence is flagged as suspicious and has to be analysed manually.

B. Passive Model Learning

The second category includes the techniques that passively recover models from a given set of samples, e.g., a set of execution traces. These are said passive as there is no direct interaction with the system under learning. Models are often generated by encoding sample sets with state diagrams whose equivalent states are merged. For instance, the k-Tail approach [3] merges the states having the same k-future, i.e., the same event sequences having the maximum length k, which all are accepted by the two states. k-Tail has been later enhanced with Gk-tail to generate Extended Finite State Machines encoding data constraints [4]. Other approaches also enhance k-Tail to build more precise models [5, 7, 8]. kBehavior [6] is another kind of approach that generates models from a set of traces by taking every trace one after the other and by completing a finite-state automaton in such a way that it now accepts the trace. These previous passive algorithms usually yield big models, which may quickly become unreadable.

Some passive approaches dedicated to communicating systems have also been proposed. Mariani et al. proposed in [19] an automatic detection of failures in log files by means of model learning. This work extends kBehavior to support events combined with data. It segments an event log with two strategies: per component or per user. The former, which can be used with communicating systems, generates one model for each component. CSight [9] is another tool specialised in the model learning of communicating systems, where components exchange messages through synchronous channels. It is assumed that both the channels and components are known. Besides, CSight requires specific trace sets, which are

segmented with one subset by component. CSight follows five stages: 1) log parsing and mining of invariants 2) generation of a concrete Finite State Machine (FSM) that captures the functioning of the whole system by recomposing the traces of the components; 3) generation of a more concise abstract FSM; 4) model refinement with invariants that must hold in FSMs, and 5) generation of Communicating FSM.

C. Key Observations and Motivations

After having studied the literature, we have firstly observed that few papers and tools tackled the model generation of component based systems or communicating systems. As stated in the introduction, the main concerns of the active model learning techniques are that the component topology must be known in advance, and that all the components must be reachable, testable and resettable many times. As a consequence, active learning can be currently applied on a limited amount of systems. As for passive techniques, the approaches [9, 19] have paved the way, however, there is still room of improvements to relax the approach requirements and to infer precise models. Besides, we have observed that the generation and use of invariants to make models more precise also limits learning to small trace sets only in practice. For instance, the invariant mining and satisfiability checking used in CSight are both costly and prevent the tool from taking as input medium to large trace sets.

We have proposed in [25] a passive model learning algorithm for component-based systems, which builds one model per component to avoid the generation of large and unreadable models. This approach is specialised to IoT systems with an algorithm called Assess [26]. The requirements considered in these approaches are different from those of CkTail or CSight. The main difference lies in the fact that the communications among components are assumed hidden (not available in event logs). Therefore, Assess tries to detect implicit component calls and adds new synchronisation actions in models. Its algorithm is hence specific to this assumption. Then, we have proposed CkTailv1 [2] to generate models of communicating systems. In short, the novelty proposed by CkTailv1 lies in its capability of detecting sessions in event logs. Indeed, CSight needs sessions put in separate sets but does not provide a way to generate them from event logs. The work proposed in [6] offers the possibility to segment event logs with several strategies. One of them allows to extract the session of every component on condition that the events include component identifiers.

We showed that CkTailv1 builds more precise models than the other approaches by better recognising sessions, but we also concluded that its requirements are too restrictive to be widely used. Indeed, CkTailv1 requires event logs comprising communication events only in such a way that each request has to be followed by one response only. CkTailv2 aims at relaxing some of these assumptions and integrates two new trace extraction algorithms to support more communicating systems.

III. CkTailv2 Tool and Approach Presentation

CkTailv2 is implemented in Java and is released as open source in [20]. The tool takes as inputs an event log collected from a communicating system and a file including regular expressions used to format the event log. It returns two kinds of models. The behaviours of each component of the system under learning are encoded with one IOLTS. Intuitively, an IOLTS expresses here the interactions of one component c with the others along with the non-communication actions of c. Besides, CkTailv2 generates dependency graphs, given under the form of Direct Acyclic Graphs (DAGs). Each component has its own DAG capturing its dependencies towards other components. Such graphs help better comprehend the architecture of the whole system. They complement the IOLTSs by offering another viewpoint of the component interactions and they might be used to different purposes, e.g., testability measurement, or security analysis. Once generated, CkTailv2 stores these models into two folders containing files saved in the DOT format. We chose this format since it is based upon a well-known plain text graph description language that can be translated into graphics formats, e.g., PDF.

We provide below the requirements of CkTailv2, an overview of its architecture and functioning along with an example of model generation.

A. CkTailv2 Requirements

The capability of CkTailv2 of inferring models depends on several realistic assumptions made on a system under learning denoted SUL:

- **A1 Event log:** we consider the components of SUL as black-boxes (no access to firmware, code, data stored on the device, etc.). The communications among the components can be monitored, e.g., on components, on servers, gateways, or by means of wireless sniffers. Event logs are collected in a synchronous environment made up of synchronous communications. Besides, these events are ordered by means of timestamps given by a global clock. At the end of the monitoring process, we consider having one event log;
- **A2 Event content:** components produce communication events or non-communication events. Both kinds of events include parameter assignments allowing to identify the source and the destination of each event. For non-communication events, both the source and the destination refer to the same component that has produced the event. Besides, a communication event can be identified either as a request or a response;
- **A3 Device collaboration:** components can run in parallel and communicate with each other. To learn precise models, we want to recognise sessions of the system in event logs. We consider two exclusive cases:
  - **A31:** the components of SUL follow this strict behaviour: they cannot run multiple instances; requests are processed by a component on a first-come, first served basis. Besides, components follow the request

–response exchange pattern (a response is associated to one request, a request is associated to one or more responses), or

– **A32:** the events that belong to the same session are identified by a parameter assignment.

The session recognition mentioned in A3 helps extract traces expressing complete behaviours of SUL, i.e., disjoint action sequences starting from one of its initial states and ending in one of its final states. A32 represents the classical assumption stating that messages include an identifier allowing to observe whole collaborations among components. Usually, the session identification strongly facilitates the trace extraction. Unfortunately, we have observed that this technique is seldom adopted with communicating systems. When it is not used, we restrict the functioning of SUL with A31 to be able to recognise sessions. We have observed that this assumption can be applied with many wireless or IoT systems.

B. CkTailv2 Overview

CkTailv2 is organised into four-steps, illustrated in Figure 1. Initially, the user gives as inputs an event log collected from SUL along with regular expressions. The latter are used to format the event log into a sequence S of actions of the form $a(\alpha)$ with a a label and α some parameter assignments. In accordance with the assumptions A1-A3, the event log formatting allows to highlight some information such as timestamps, or the sources and destinations of the messages (request or response).

Execution traces are extracted from S by means of two algorithms, which rely either on the assumption A31 or A32.

In short, if the actions include session identifiers, allowing to directly recognise sessions in S (A32), The first algorithm which aims at recognising sessions in S with respect to constraints derived from the assumptions A1-A31. The second one extracts traces by using session identifiers. When these identifiers are provided, the algorithm simply extracts traces by covering actions and their respective identifiers. When the latter are unknown, the event log is analysed w.r.t. session patterns and quality metrics to recover these identifiers first. Both algorithms return a trace set denoted $Traces(\mathrm{SUL})$ and detect dependencies among the components of SUL. Then, the third step of CkTailv2 derives dependency graphs from $Deps(\mathrm{SUL})$. In its last step, CkTailv2 generates one IOLTS for every component of SUL with three sub-steps called "4A Trace partitioning", "4B IOLTS Generation" and "4C IOLTS Generalisation". The latter calls the k-Tail approach, which is a model learning technique used to reduce IOLTSs by merging equivalent states.

C. Model Learning Example

Before describing the CkTailv2's steps, let us illustrate them with a motivating example of model generation. Figure 2 shows a part of an event log collected from an IoT system made up of devices and of two gateways. The events are formatted by means of regular expressions to produce actions. The regular expression example of Figure 2 extracts from HTTP requests a label equals to the URI along with some parameters. Figure 3 depicts an example of sequence of 15 actions obtained after the first step of CkTailv2. The first four actions are derived from the HTTP messages of Figure 2. As required, these actions indicate the sources and destinations of the messages with the parameters $from$ and to. The other parameter assignments capture acknowledgements or sensor data, e.g., a temperature value with svalue:=68 or a level of luminance with svalue:=1000. We can observe from these actions that the IoT system SUL is made up of 6 components. But interpreting their interactions and what they do is still tricky because of lack of readability.

Traces are now extracted from the action sequence S of Figure 3 by the second step of CkTailv2. It covers and segments S while trying to recognise sessions. In our example, no session identifier is found in the actions. As a consequence, CkTailv2 uses an algorithm that tries to recover sessions with respect to the assumption A31. To be integrated in the algorithm, we formulated this assumption with five constraints expressing what a session is and when keeping an action to a current session. These constraints are detailed in Section IV-C and summarised as follows: C1: a response is always associated to the last observed request sharing the same communicating components; C2: successive responses are always associated to the related request; C3: nested requests (a request to a component that also performs another request before giving a response) are always kept together in a session; C4: a session gathers messages exchanged between components interacting together in a limited time delay and all the messages capturing a data dependency between two components; C5: a non-communication event is kept is the current session also with respect to time delay and data dependency. Figure 4 gives the trace set $Traces(\mathrm{SUL})$ obtained from the action sequence of Figure 3 with this algorithm. For sake of readability, the parameter assignment are concealed in the figure. We observe that it has kept together the related requests and responses, and the nested requests req6 req7. Here, our algorithm has only detected one distinctive longer time interval between the two actions resp5 req6, which implicitly shows that a session ends at resp5 and that a new one begins at req6.

While actions are covered to extract traces, the component interactions are also analysed by CkTailv2 for detecting component dependencies. These dependencies are given under the form of component lists $c_1c_2 \dots c_k$ expressing that a component c_1 depends on a component c_2, which itself depends on another component and so on. The set $Deps(\mathrm{SUL})$ gathers these component lists. The component dependency is defined in Section IV-E. Figure 4 shows the set $Deps(\mathrm{SUL})$ inferred from our example. Most of the dependencies between pairs of components stem from requests. The component sequence G1G2d3 is detected from the nested requests req6 req7. Four data dependencies are also detected between d2d1, G2d1, d4d1, (with the data svalue:=68) and d3G1 (with the data cmd:=status).

The CkTail's third step generates dependency graphs. It derives DAGs from the set $Deps(\mathrm{SUL})$ and computes their

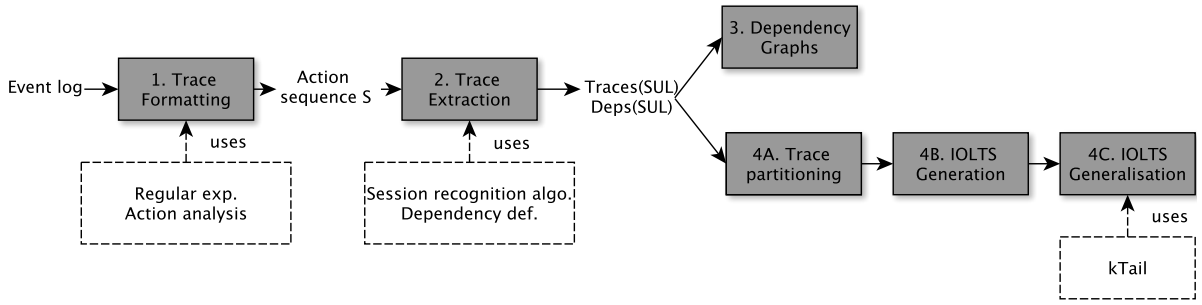


Fig. 1. Model learning of communicating systems with the CkTail approach

```
Jan 20, 2020 09:56:24.225
    CET;Host=d1;Dest=G1;Protocol=HTTP;Verb=GET
    Uri=/req1?svalue=68.00 HTTP/1.1;
Jan 20, 2020 09:56:24.682
    CET;Host=G1;Dest=d1;Protocol=HTTP;HTTP/1.1
    status=200 response=OK;
Jan 20, 2020 09:56:25.153
    CET;Host=G1;Dest=d2;Protocol=HTTP;Verb=GET
    Uri=/req2?svalue=68.00 HTTP/1.1;
Jan 20, 2020 09:56:25.318
    CET;Host=d2;Dest=G1;Protocol=HTTP;HTTP/1.1
    status=200 response=OK data=done;

Example of regular expression:
^(?<date>\w{3} \d{2}, \d{4} \d{2}:\d{2}:\d{2}.\d{3})
\s(CET);(?<param1>(\w+=\w\d));(?<param2>(\w+=\w\d));
(?<param3>[^;]+);(?<param4>[^=]+=[A-Z]{3,4})\s(Uri=)
(?<label>[^?]+)[?](?<param5>(\w+=\d{2}\.\d{2}))\s
HTTP/1.1;$
```

Fig. 2. Example of 4 HTTP messages collected from an IoT system. The regular expression retrieves a label and 5 parameters here. The label expression will be the label of the action in the action sequence S

transitive closures. Figure 5 illustrates the dependency graphs obtained in our example.

The fourth step of CkTailv2 lifts traces to the level of IOLTSs. In the step *4A Trace partitioning*, CkTailv2 builds one trace set for every component of SUL. It begins by doubling every communication action to give a pair of output/input actions by separating the notion of source/destination. The non-communication actions are marked as outputs. $Traces(\text{SUL})$ is then partitioned into as many trace sets as components found in SUL. Each trace set T_c gathers only the traces related to the component c. If we take back our example, Figure 6 gives the new trace sets composed of sequences of input and output actions derived from the set $Trace(\text{SUL})$ of Figure 4. As this system is made up of 6 components, $Traces(\text{SUL})$ is partitioned into 6 subsets.

The step *4B IOLTS Generation* transforms every trace set T_c into an IOLTS by converting traces into IOLTS path cycles, which are joined on the initial state only. In our example, as we have 6 trace sets, we obtain 6 IOLTSs Ld1-Ld4, LG1, LG2, illustrated in Figure 7. Finally, CkTailv2 applies the k-Tail algorithm to reduce the IOLTS sizes in the step *4C IOLTS Generalisation*. More precisely, it merges the states sharing

```
req1(from:=d1,to:=G1,svalue:=68,time:=
09:56:24.225)
resp1(from:=G1,to:=d1,content:=ok, time:=
09:56:24.682)
req2(from:=G1,to:=d2,svalue:=68, time:=
09:56:25.153)
resp2(from:=d2,to:=G1,content:=done,
time:=09:56:25.318)
req3(from:=G1,to:=G2,svalue:=68, time:=
09:56:26.267)
req1(from:=d1,to:=G1,svalue:=68, time:=
09:56:27.369)
resp3(from:=G2,to:=G1,content:=ok, time:=
09:56:27.371)
resp1(from:=G1,to:=d1,content:=ok, time:=
09:56:27.720)
req5(from:=G2,to:=d4,svalue:=68, time:=
09:56:27.859)
log(from:=d4,to:=d4,content:=heat-off,
time:=09:56:28.909))
resp5(from:=d4,to:=G2,content:=done,
time:=09:56:28.982)
req6(from:=G1,to:=G2,udevice:=12, cmd:=
status,time:=09:56:35.962)
req7(from:=G2,to:=d3,cmd:=status,GPIO:=1
time:=09:56:35.974)
resp7(from:=d3,to:=G2,svalue:=1000,
time:=09:56:36.846)
resp6(from:=G2,to:=G1,svalue:=1000, time:=
09:56:36.958)
```

Fig. 3. Action sequence of an IoT system. These actions have the form <label><parameter assignments>, the latter expressing the components involved in the communications and data

```
Traces(SUL)={req1resp1req2resp2req3req1resp3
resp1req5logresp5, req6req7resp7resp6}

Deps={ d1G1, G1d2, d2d1, G1G2, G2d4, G2d1,
    d4d1, G1G2d3, G2d3, d3G1 }
```

Fig. 4. Step 2: Traces of SUL and dependency set $Deps(\text{SUL})$. The parameter assignments are concealed for readability

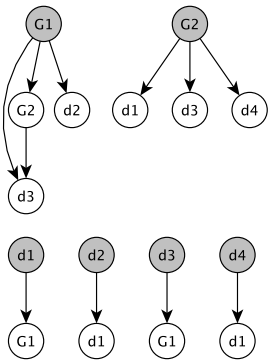


Fig. 5. Step 3: Dependency Graph Generation. Each component has its own dependency graph expressing its directional dependencies with the other components of SUL

```
$T_{d1}$={ !req1 ?resp1 !req1 ?resp1 }
$T_{d2}$={ ?req2 !resp2 }
$T_{d3}$={ ?req7 !resp7 }
$T_{d4}$={ ?req5 !log !resp5 }
$T_{G1}$={ ?req1 !resp1 !req2 ?resp2 !req3
    ?req1 ?resp3 !resp1, !req6 ?resp6 }
$T_{G2}$={ ?req3 !resp3 !req5 ?resp5, ?req6
    !req7 ?resp7 !resp6 }
```

Fig. 6. Step 4A: Trace Partitioning. $Traces$(SUL) is prepared before the IOLTS generation. $Traces$(SUL) is segmented to produce one trace set for every component of SUL

the same k-future, i.e., the same action sequences having the maximum length k. In our example, with $k := 2$, only the states in white of the IOLTS Ld1 are merged by k-Tail, which produces the IOLTS Ld11.

With these IOLTSs and DAGs, it becomes easier to interpret the behaviour of SUL. In our example, the IOLTSs bring out that the central devices of SUL are G1 and G2, which are the two gateways. The component $d1$ is an active sensor that provides temperature values. These values are sent to two actuators d2 and d4 through the gateways G1 and G2. d3 is a passive sensor (an illuminance light meter) that is queried by G1 through G2, as d3 is directly connected to G2. d4 seems to control a heating system, which is turned off when the temperature reaches 68°F.

Furthermore, as we now have formal models, different kinds of activities may be automatically or semi-automatically conducted to document SUL, to discover defects or more generally to audit SUL. For instance, the European Telecommunications Standards Institute (ETSI) has proposed a general method dedicated to audit large scale, networked systems by undertaking testing and risk assessments [27]. One of the stages of this method corresponds to establishing the context of SUL, which can be partially performed with our tool from event logs. Besides, quality metrics such as testability degrees can be computed from our models [28, 29]. We provide another tool for computing Observability, Controllability and Dependability in [30]. These metrics can be used to deduce which component is testable, or testable in isolation. Other approaches can take these models or transition systems to audit the security of SUL [24, 31, 32, 33].

After this overview of CkTailv2, we will develop its theoretical background along with its algorithms in the next section.

IV. The CkTailv2 Approach

Before going to the CkTailv2 step description, we will briefly recall some basic definitions and notations used in the remainder of the paper.

A. Preliminary Definitions

As in many works dealing with the modelling of atomic components, e.g., [34, 35], we express the behaviours of components with the well established Labelled Transition System (LTS) model. A LTS is defined in terms of states and transitions labelled by actions, themselves taken from a general action set $\mathcal{L}$, which expresses what happens. The Input Output LTS is an extension of the LTS allowing to better express behaviours with inputs and outputs.

Definition 1 (IOLTS) *An Input Output Labelled Transition System (IOLTS) is a 4-tuple $\langle Q, q0, \Sigma, \rightarrow \rangle$ where:*

- *Q is a finite set of states;*
- *$q0$ is the initial state;*
- *$\Sigma \subseteq \mathcal{L}$ is the finite set of actions. $\Sigma_I \subseteq \Sigma$ is the countable set of input actions, $\Sigma_O \subseteq \Sigma$ is the countable set of output actions, with $\Sigma_O \cap \Sigma_I = \emptyset$;*
- *$\rightarrow \subseteq Q \times \Sigma \times Q$ is a finite set of transitions. A transition (q, a, q') is denoted $q \xrightarrow{a} q'$.*

We also use the following notations on action sequences. The concatenation of two action sequences σ_1, $\sigma_2 \in \mathcal{L}^*$ is denoted $\sigma_1.\sigma_2$. ϵ denotes the empty sequence. We denote that σ_1 is a subsequence of another sequence σ_2 with $\sigma_1 \preceq \sigma_2$. $final(\sigma)$ denotes the action $a_k(\alpha_k)$ of the sequence $\sigma = a_1(\alpha_1) \dots a_k(\alpha_k)$ or ϵ if $\sigma = \epsilon$. A trace is a finite sequence of observable actions in $\mathcal{L}^*$. The trace of an IOLTS is a sequence $a_0 \dots a_k$ such that $\exists q_{i-1}, q_i, a_i, (1 \leq i \leq k) : q0 \xrightarrow{a_1} q_1 \dots q_{k-1} \xrightarrow{a_k} q_k \in \rightarrow^*$. $Traces(L)$ denote the trace set of the IOLTS L.

Furthermore, to better match the functioning of communicating systems, we assume that an action has the form $a(\alpha)$ with a a label and α an assignment of parameters in P, with P the set of parameter assignments. For example, $!switch(from := c_1, to := c_2, cmd := on)$ is an output action composed of the label "switch" followed by a parameter assignment expressing the components involved in the communication and a parameter of the switch command.

We will finally use the following notations on actions to make our algorithms more readable:

- $from(a(\alpha)) = c$ denotes the source of the action;
- $to(a(\alpha)) = c$ denotes the destination;
- $components(a(\alpha)) = \{from(a(\alpha)), to(a(\alpha))\}$;

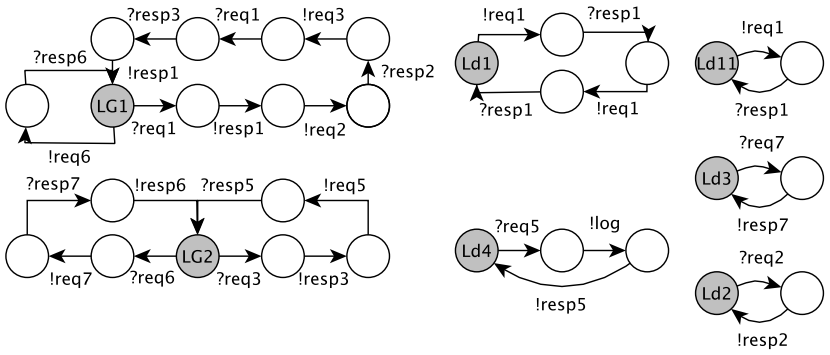


Fig. 7. Steps 4B & C: IOLTS Generation and generalisation with the k-Tail algorithm. Each component has its own IOLTS.

- $time(a(\alpha)) = t$ returns the timestamp value identifying when $a(\alpha)$ occurred, $time(\epsilon) = +\infty$;
- $isReq(a(\alpha))$, $isResp(a(\alpha))$ are boolean expressions expressing the nature of the message;
- $session(a(\alpha)) = id$ denotes the session identifier when available. Otherwise, $session(a(\alpha)) = \emptyset$.
- $data(a(\alpha)) = \alpha \setminus \{from := c_1; to := c_2, time := t, session := s\}$;

The dependencies among the components of a communicating system are captured with a Directed Acyclic Graph (DAG), where component identifiers are labelled on vertices.

Definition 2 (Directed Acyclic Graph) *A DAG Dg is a 2-tuple $\langle V_{Dg}, E_{Dg} \rangle$ where V is the finite set of vertices and E the finite set of edges.*
λ denotes a labelling function mapping each vertex $v \in V$ to a label $\lambda(v)$.

B. CkTailv2 Step 1: Trace Formatting

Keeping in mind the assumption A1, CkTail takes as input an event log gathering events that are totally ordered by means of their time-stamps. These events are parsed to retrieve the actions performed by SUL and their related data. These actions must have the form $a(\alpha)$ with a a label and α an assignment of parameters and must be compliant with the assumption A2. This formatting is achieved by means of regular expressions given to CkTailv2. Their writing may be performed manually with small to medium event logs, but this activity may quickly become laborious as the log size grows. A way to eliminate or assist users in this intervention is to consider the approaches and tools that automatically mine patterns from log files [19, 36, 37, 38, 39, 40]. These patterns may be used to quickly derive regular expressions.

As events are usually too detailed or specific to their related executions, regular expressions are also a good mean to lift the abstraction level by filtering out some useless actions, or some concrete values in actions.

At the end of this step, we hence assume having a sequence $S \in \mathcal{L}^*$ of actions on the form $a_1(\alpha_1) \dots a_k(\alpha_k)$. The next step of CkTailv2 covers the action sequence S to extract the sub-sequences that capture some sessions of SUL. This step relies either on the assumption A31 or A32 and is hence implemented with two different algorithms presented in the two next sections.

TABLE I
CONSTRAINTS DERIVED FROM THE ASSUMPTIONS A1, A2, A31. WHEN ONE OF THESE CONSTRAINTS HOLD, THE CURRENT ACTION $a_i(\alpha_i)$ IS KEPT IN A SESSION σ.

C1	A response $a_i(\alpha_i)$ is always associated to the last request previously observed in σ such that the replier returns the response to the requester which has sent the request.
C2	All the responses associated to the same request are kept in σ.
C3	A request $a_i(\alpha_i)$ that belongs to a chain of nested requests must be kept in the session σ. Two requests req1 and req2 are nested iff the action sequence S includes this form of sequence: req1(from:=c1, to:=c2) req2(from:=c2, to:=c3) resp2(from:=c3, to:=c2) resp1(from:=c2, to:=c1).
C4	A component, which already participated to the session σ, can send a new request $a_i(\alpha_i)$ to another component. This request is kept in σ if C4.1: the session is not timed out, or if C4.2: this request shares data with some previous actions of σ
C5	A non-communication action $a_i(\alpha_i)$ is kept in σ if C5.1: the session is not timed out, or if C5.2: $a_i(\alpha_i)$ shares data with some previous actions of σ

C. CkTailv2 Step 2: Trace Extraction Without Session Identifier

The first trace extraction algorithm is founded on the assumptions A1, A2, A31 to interpret communications and to recover sessions in event logs. In particular, with A31, we suppose that sessions are not identified in event logs.

To devise this algorithm, we derived a list of constraints from these assumptions giving the conditions for a sub-sequence of S to be a session. As our algorithms cover the actions of S one after the other, we have formulated these constraints to express whether an action $a_i(\alpha_i)$ of the action sequence $S = a_1(\alpha_1) \dots a_k(\alpha_k) \in \mathcal{L}^*$ belongs to a session denoted σ. Table I gathers the five constraints used in our algorithms. C1 and C2 focus on responses, while C3 and C4 deal with requests. C4 is a special constraint expressing when a new request $a_i(\alpha_i)$, sent from a component that has previously participated in the current session, belongs to σ. The choice of keeping this new request in the session depends on two other factors, i.e., time delay and data dependency, with the constraints C4.1 and C4.2. C5 addresses non-communication actions and restricts the session participation as in C4.

TABLE II
FORMALISATION OF THE CONSTRAINTS C1-C5 USED IN THE TRACE EXTRACTION ALGORITHM

C1	$\exists!\sigma_r \in Lreq(\sigma) : response(a_i(\alpha_i), final(\sigma_r))\}$
C2	$\exists!\sigma_r \in OLreq(\sigma) : response(a_i(\alpha_i), final(\sigma_r))\}$
C3	$isReq(a_i(\alpha_i)) \wedge Lreq' = \{\sigma_1 \in Lreq(\sigma) \mid from(a_i(\alpha_i)) = to(final(\sigma_1))\} \neq \emptyset \wedge \neg pendingRequest(from(a_i(\alpha_i)))$
C4	$isReq(a_i(\alpha_i)) \wedge$ $from(a_i(\alpha_i)) \in \text{KC} \wedge (\forall\sigma_1 \in Lreq(\sigma) : from(a_i(\alpha_i)) \neq to(final(\sigma_1))) \wedge (ontime(a_i(\alpha_i), \sigma) \vee dataDependency(a_i(\alpha_i), S, \sigma)) \wedge \neg pendingRequest(from(a_i(\alpha_i)))$
C5	$\neg isReq(a_i(\alpha_i)) \wedge \neg isResp(a_i(\alpha_i)) \wedge from(a_i(\alpha_i)) \in \text{KC} \wedge (ontime(a_i(\alpha_i), \sigma) \vee dataDependency(a_i(\alpha_i), S, \sigma))$

To use these constraints in our algorithms, we formulated them with boolean expressions written with the notations given in Section IV-A completed by these ones:

- KC stands for the set of known components involved in the session σ so far;
- $response(a_1(\alpha_1), a(\alpha))$ is the boolean expression $isResp(\ a_1(\alpha_1)) \wedge\ from(a_1(\alpha_1)) = to(a(\alpha)) \wedge\ to(a_1(\ \alpha_1)) = from(a(\alpha))$;
- $Lreq(\sigma)$ denotes the set of sequences of pending requests i.e., the sequences of requests $a_1(\alpha_1) \dots a_k(\alpha_k) \preceq \sigma$ for which responses have not yet been received. $Lreq(\sigma) =_{def} \{a_1(\alpha_1) \dots a_k(\alpha_k) \preceq \sigma \mid isReq(a_i(\ \alpha_i))_{1 \leq i \leq k}, \forall a(\alpha) \in \mathcal{L}^* : response(a(\alpha), a_i(\alpha_i)) \implies a_i(\alpha_i)a(\alpha)a_{i+1}(\alpha_{i+1}) \not\preceq \sigma\}$;
- $OLreq(\sigma)$ denotes the set of requests for which a least one response has been received;
- $ontime(a(\alpha), \sigma)$ is a boolean expression that returns true if the action $a(\alpha)$ may belong to the session σ with regard to the session duration or session time-out;
- *data-dependency*$(a(\alpha), S, \sigma)$ is a boolean expression that returns true if the request $a(\alpha)$ shares some data with other requests of the session $\sigma \preceq S$. The data dependency is defined in Section IV-E;
- $pendingRequest(c)$ is the boolean expression $(\exists\sigma_1 \in Lreq(\sigma), a(\alpha) \in \sigma_1 : c \in components(a(\alpha)))$ that evaluates whether the component c has sent (resp. received) a request and has not yet received (resp. sent) the response.

From these notations, we formulated the above constraints, listed their boolean terms and studied their possible permutations. We finally kept the constraints expressing that an action $a_i(\alpha_i)$ belongs to the current session when they hold. These are listed in Table II.

Algorithms 1 and 2 implement the trace extraction. Algorithm 1 calls the procedure ***Keep-or-Split*** with an action sequence initialised to S. It returns $Traces(\text{SUL})$, the final component set C along with the set of component dependencies $Deps(\text{SUL})$.

The procedure ***Keep-or-Split*** covers an action sequence $a_1(\ \alpha_1) \dots a_k(\alpha_k)$ to extract a session σ. The set of known components KC is initialised with the components of the first action $a_1(\alpha_1)$. Then, every action $a_i(\alpha_i)$ is covered to decide whether it is kept in σ (line 8) or not. Given an action $a_i(\alpha_i)$, the procedure $updateOLreq$ (Algorithm 2 lines (1-5)) is called to update the set of pending requests $OLreq$ w.r.t. the assumption A31. More precisely, if $a_i(\alpha_i)$ is a new request coming from a component c, then all the previous requests that involve c are removed from $OLreq$ to meet A31 (first come, first served). In the same way, if $a_i(\alpha_i)$ is a response, only the request associated to this response is kept.

Algorithm 1: Trace Extraction with A31

```
   input  : Action sequence S
   output : Traces(SUL), Component set C, Component dependency set
            Deps(SUL)
1  C := Deps(SUL) := ∅;
2  Keep-or-Split(S);
3  Procedure Keep-or-Split(a_1(α_1) ... a_k(α_k)) is
4      σ := σ_2 := ϵ;
5      Lreq(σ) := OLreq(σ) := ∅;
6      KC := components(a_1(α_1));
7      i := 1;
8      while i ≤ k do
9          updateOLreq(a_i(α_i));
10         case C1 true do
11             σ := σ.a_i(α_i); Trim(σ_r);
12             KC := KC ∪components(a_i(α_i));
13         case C1 false and C2 true do
14             σ := σ.a_i(α_i);
15             KC := KC ∪components(a_i(α_i));
16         case C3 true do
17             σ := σ.a_i(α_i);
18             Extend(σ_r, a_i(α_i));
19             KC := KC ∪components(a_i(α_i));
20         case C3 false and C4 true do
21             σ := σ.a_i(α_i);
22             Extend(ϵ, a_i(α_i));
23             KC := KC ∪components(a_i(α_i));
24         case C5 true do
25             σ := σ.a_i(α_i);
26             KC := KC ∪components(a_i(α_i));
27         otherwise do σ_2 := σ_2.a_i(α_i) ;
28         i++;
29     Traces(SUL) := Traces(SUL) ∪ {σ};
30     C := C ∪ KC;
31     if σ_2 ≠ ϵ then
32         Keep-or-Split(σ_2);
33 END;
```

Then, the procedure ***Keep-or-Split*** processes the action $a_i(\alpha_i)$ with the constraints C1-C5. When one of them holds, the action $a_i(\alpha_i)$ is added to the session σ. Besides, the set of known components KC is updated to include the components involved in $a_i(\alpha_i)$. For any other case, the action $a_i(\alpha_i)$ is put into a new action sequence σ_2 (line 27). Once all the actions have been covered, σ is added to $Traces(\text{SUL})$ and C is updated with the set of components KC built with this session. If σ_2 is not empty, the procedure ***Keep-or-Split(σ_2)*** is recursively called to recover other sessions in σ_2 (line 31).

The main difference among the cases C1 to C5 lies in the update of the set of pending requests $Lreq(\sigma)$, with the procedures $Trim$ and $Extend$. The former is called with C1: receipt of a response associated to a list of pending requests σ_r in $Lreq(\sigma)$. $Trim$ is called to remove the last request of σ_r, $final(\sigma_r)$, because a response has been received to this request. $final(\sigma_r)$ is shifted to $OLreq(\sigma)$. The procedure $Extend$ is called with C3 and C4. C3 corresponds to the receipt of a request that belongs to a chain of nested requests $\sigma_r \in Lreq(\sigma)$. $Extend$ is here called to update $Lreq(\sigma)$ with

the nested request list $\sigma_r.a_i(\alpha_i)$. C4 stands for the receipt of a new request from a known component. $Extend$ is now called to add the new request $a_i(\alpha_i)$ in $Lreq(\sigma)$. Furthermore, $Extend$ builds the set $Deps(\text{SUL})$ of component lists. This part is detailed in Section IV-E.

Algorithm 2:

```
1  Procedure updateOLreq(a_i(α_i)) is
2      if isReq(a_i(α_i)) then
3          OLreq(σ) := OLreq(σ) \ {a(α) ∈ OLreq |
             from(a_i(α_i)) ∈ components(a(α))};
4      else if isResp(a_i(α_i)) then
5          Lr := {a(α) ∈ OLreq(σ) | from(a_i(α_i) = to(a(α))}
           OLreq(σ) := OLreq(σ) \ {a(α) ∈ OLreq(σ) |
             from(a_i(α_i) ∈ components(a(α))} ∪ Lr;
6  Procedure Trim(σ_r) is
7      σ' := remove(final(σ_r));
8      Lreq(σ) := Lreq(σ) \ {σ_r} ∪ {σ'};
9      OLreq(σ) := OLreq(σ) \ {a(α) ∈ OLreq(σ) |
         from(final(σ_r)) ∈ components(a(α))};
10     OLreq(σ) := OLreq(σ) ∪ {final(σ_r)};
11 Procedure Extend(σ_r, a(α)) is
12     σ' := σ_r.a(α) = a_1(α_1) ... a_k(α_k);
13     Lreq(σ) := Lreq(σ) \ {σ_r} ∪ {σ'};
14     //Component dependencies
15     lc := c_1 ... c_k c_{k+1} such that c_i = from(a_i(α_i))_(1≤i≤k),
         c_{k+1} = to(a_k(α_k));
16     Deps(SUL) := Deps(SUL) ∪ {lc};
17 Procedure ontime(a_i(α_i), σ) is
18     return (time(a_i(α_i)) − time(final(σ)) < T);
19 Procedure data-dependency(a_i(α_i), S, σ) is
20     if ∃σ_1 = a_1(α_1)a_2(α_2) ... a_i(α_i) ⪯ S : to(a_i(α_i)) --data--> σ_1
         from(a_1(α_1)) then
21         Deps(SUL) := Deps(SUL)∪{to(a_i(α_i)).from(a_1(α_1))};
22         if σ_1 ⪯ σ.a_i(α_i)) then
23             return true;
24     return false;
```

The boolean expression $ontime(a(\alpha), \sigma)$, used in C4 and C5, is implemented with the procedure $ontime$. As stated previously $ontime$ allows to limit the session duration. Several implementations are possible. We provide an example in Algorithm 2, line (17). This procedure checks whether the time delay between the last received action $a_i(\alpha_i)$ and the previous one in the session σ is lower than a time duration T.

The boolean expression *data-dependency* $(a_i(\alpha_i), S, \sigma)$, also used in C4 and C5, is implemented by the procedure given in Algorithm 2. It checks whether a data dependency exists between the request $a_i(\alpha_i)$ and some requests of the session σ. The notion of dependency among components and this procedure shall be discussed in Section IV-E.

The action sequence of Figure 3 has been converted into $Traces(\text{SUL})$ by means of this algorithm, as no session identifier is available within actions. Here, the trace extraction algorithm has detected that C4 does not hold with the request req6. It has indeed detected, by means of the timestamps, a distinctive longer time interval between the actions resp5 req6, which implicitly suggests that the session timed out. The algorithm has detected two nested requests req6 req7. Besides, several data dependencies have been identified between the requests req1,req2 req3,req5. These requests along with their responses are hence kept together in the same session.

D. CkTailv2 Step 2: Trace Extraction With Session Identifiers

Algorithm 3: Trace Extraction with A32

```
   input : Action sequence S
   output: Traces(SUL), Component set C, Component dependency set
           Deps(SUL)
1  C := Deps(SUL) := ∅;
2  ID := {session(a(α)) | a(α) ∈ S};
3  Traces(SUL) := ⋃_{id∈ID} {σ_id} with
     σ_id = S \ {a(α) | session(a(α)) ≠ id};
4  foreach σ = a_1(α_1) ... a_k(α_k) ∈ Traces(SUL) do
5      S := σ;
6      Keep-or-Split2(S);
7  END;
8  Procedure Keep-or-Split2(a_1(α_1) ... a_k(α_k)) is
9      Lreq(σ) := OLreq(σ) := ∅;
10     KC := components(a_1(α_1));
11     i := 1;
12     while i ≤ k do
13         C := C ∪ components(a_i(α_i));
14         case C1 true do
15             Trim(σ_r);
16         case C3 true do
17             Extend(σ_r, a_i(α_i));
18         case C3 false and C4 true do
19             Extend(ϵ, a_i(α_i));
20         i++;
```

The previous trace extraction algorithm relies on the assumption A31 to extract traces. This second trace algorithm now relies on A32. This assumption involves that the session identifiers should be given to the algorithm. Nonetheless, we observed that establishing an identifier list is a strong assumption especially when SUL is a composition of external items, e.g., services or IoT, whose functioning is not known.

To solve this issue, we presented in [1] an algorithm for extracting session identifiers from event logs. In short, this algorithm explores the trace set space that can be derived from an event log along with the respective identifiers. Furthermore, it is guided toward the most relevant solutions by means of session invariants and trace quality metrics. The algorithm either provides a first session identifier set that meets quality or returns a sorted list w.r.t. quality. More details were presented in [1].

With a given set of session identifiers and A32, the trace extraction is quite simpler to perform. Algorithm 3 begins to build $Traces(\text{SUL})$ by extracting from S the sub-sequences of actions having the same session identifier (lines 2-3). Afterwards, it calls the procedure *Keep-or-Split2* for every trace of $Traces($ SUL$)$ to detect component dependencies as previously (lines 4-6). To this end, this procedure updates the set of pending requests $Lreq(\ \sigma)$ as previously for every trace σ with the constraints C1, C3, C4 (only these constraints are needed to build $Lreq(\sigma)$). $Lreq(\sigma)$ is updated by means of the procedures $Trim$ and $Extend$, which disclose component dependencies and build the set $Deps(\text{SUL})$.

E. CkTailv2 Step 3: Dependency Graph Generation

The notion of component dependency is formulated by means of the three expressions given below. We write c_1 *depends on* c_2, when at least one of these expressions holds.

Definition 3 (Component dependency) *Let* $c_1, c_2 \in C$, $c_1 \neq c_2$, *and* $S \in \mathcal{L}^*$. *We denote* c_1 *depends on* c_2 *iff* $(c_1 \xrightarrow[\sigma]{r} c_2) \vee (c_1 \xrightarrow[\sigma]{nr} c_2) \vee (c_1 \xrightarrow[\sigma]{data} c_2)$ *with:*

1) $c_1 \xrightarrow[\sigma]{r} c_2$ *iff* $\exists \sigma \preceq S, a(\alpha) \preceq \sigma : isReq(a(\alpha))$, $from(a(\alpha)) = c_1, to(a(\alpha)) = c_2$;
2) $c_1 \xrightarrow[\sigma]{nr} c_2$ *iff* $\exists \sigma \preceq S$, $a_1(\alpha_1) \dots a_k(\alpha_k) \preceq \sigma : from(a_1(\alpha_1)) = c_1, to(a_k(\alpha_k)) = c_2$, $a_1(\alpha_1) \dots a_k(\alpha_k) \in Lreq(\sigma)$;
3) $c_1 \xrightarrow[\sigma]{data} c_2$ *iff* $\exists \sigma \preceq S, \alpha \in P : DS(\sigma, c_1, c_2, \alpha)$ *and* $\forall \sigma' = a'_1(\alpha'_1)a'_2(\alpha'_2) \dots a_k(\alpha_k) \preceq S : DS(\sigma', c_1, c_2, \alpha) \implies \sigma' \preceq \sigma$, *with* $DS(a_1(\alpha_1) \dots a_k(\alpha_k)c_1, c_2, \alpha)$ *the boolean expression* $from(a_1(\alpha_1)) = c_2 \wedge to(a_k(\alpha_k)) = c_1 \wedge isReq(a_k(\alpha_k)) \wedge to(a_i(\alpha_i)) = from(a_{i+1}(\alpha_{i+1}))_{1 \leq i < k)} \wedge \bigcap_{(1 \leq i \leq k)} \alpha_i = \alpha$.

The two first expressions illustrate that a component c_1 depends on another component c_2 when c_1 queries c_2 with a request or by means of successive nested requests of the form $req1(from := c_1, to := c)req2(from := c, to := c_2)$. The last expression deals with data dependency. We say that c_1 depends on c_2 if there is a chain of actions from c_2 ended by a request to c_1 sharing the same data α. More precisely, the third expression holds if a component c_2 has sent an action $a_1(\alpha_1)$ with some data α, if there is a unique sequence $a_1(\alpha_1) \dots a_k(\alpha_k)$ sharing this data and if $a_k(\alpha_k)$ is a request whose destination is c_1. An immediate consequence of this expression is that we do not consider component dependencies when there are several chains of actions all sharing the same data and addressed to the several components. Yet, we can observe that there is a data dependency among components, but we are unable to establish the dependency relations as several options among the components are possible. Because of this ambiguity that may bring false relationships, we prefer to not consider this case.

The component dependencies are detected by the second step of CkTailv2 and are given under the form of component lists $c_1 \dots c_k$ in $Deps(\text{SUL})$. Component dependencies are detected while Algorithms 1 or 3 build traces by means of the procedures $Extend$ and *data-dependency*. The procedure $Extend$ detects the two first component dependency cases of Definition 3. It uses the set of pending requests $Lreq(\sigma)$ to complete the set $Deps(\text{SUL})$. Indeed, the procedure *Extend* constructs a sequence of $Lreq(\sigma)$ in such a way that it is either one request (Case C4) or a list of nested requests (Case C3). The procedure covers the component sequences $lc = c_1 \dots c_k c_{k+1}$ of $Lreq(\sigma)$ and adds the dependency lists in $Deps(\text{SUL})$ (Algorithm 2, line 15). The procedure *data-dependency*$(a_i(\alpha_i), S, \sigma)$ checks whether the last expression of Definition 3 holds. If there is a unique sequence $a_1(\alpha_1) \dots (a_i(\alpha_i)$ sharing the same data $\alpha \in data(\ a_i(\alpha_i))$ and finished by the request $a_i(\alpha_i)$ then the dependency $to(a_i(\alpha_i)).from(a_1(\alpha_1))$ is added to $Deps(\text{SUL})$ (line 21). If this sequence is a subsequence of the current session $\sigma.a_i(\alpha_i)$, then the procedure also returns true to Algorithms 1 and 3 to indicate that this request must be kept in the current session.

It is worth noting that Algorithms 1 and 3 slightly differ in the data dependency detection. Given two components c_1 and c_2, Algorithm 1 checks whether $c_1 \xrightarrow[\sigma]{data} c_2$ holds on the initial action sequence S. It checks that there is a unique chain of actions from c_2 to c_1 in S as it does not know the sessions in advance. Algorithm 3 does the same verification but on every trace σ of $Traces(\text{SUL})$, which represent sessions. As a trace σ is usually much shorter than the action sequence S, $c_1 \xrightarrow[\sigma]{data} c_2$ may be satisfied more frequently. In other terms, Algorithm 3 may detect more component dependencies because the sessions are already given and known.

Figure 4 shows the set $Deps(\text{SUL})$ derived from the action sequence of Figure 3. Most of the component dependencies stem from requests. For instance, the component sequence G1G2d3 is detected from the nested requests req6 req7. Four data dependencies are detected between d2d1, G2d1 d4d1, (with the data svalue:=68) and d3G1 (with cmd :=status).

CkTailv2 implements the generation of dependency graphs from $Deps(\text{SUL})$ with Algorithm 4. The latter partitions $Deps(\ \text{SUL})$ to group the dependency lists starting by the same component into the same subset. This partitioning is performed with the equivalence relation $\sim_c$ on C^* given by $\forall l_1, l_2 \in Deps(\text{SUL})$, with $l_1 = c_1 \dots c_k$, $l_2 = c'_1 \dots c'_k$, $l_1 \sim_c l_2$ iff $c_1 = c'_1$. Given a partition C_i and a component list $l \in C_i$, Algorithm 4 builds a path of the DAG Dg_i such that the nth state is labelled by the nth component of l. Algorithm 4 finally computes the transitive closure of the DAGs to make all component dependencies visible.

The dependency graphs, which are generated from the set $Deps(\text{SUL})$ of Figure 4, are depicted in Figure 5. They reflect another window on the architecture of SUL. Indeed, these graphs show in a readable manner how the components interact together. They also help identify central components that might have a strong negative impact on SUL when they integrate faults.

Algorithm 4: Device Dependency Graphs Generation

```
   input  : Deps(SUL)
   output : Dependency graph set DG
1  foreach C_i ∈ Deps(SUL)/ ∼_c do
2      foreach c_1 c_2 ... c_k ∈ C_i do
3          add the path s_{c_1} → s_{c_2} ... s_{c_{k-1}} → s_{c_k} to Dg_i;
4      Dg'_i is the transitive closure of Dg_i;
5      DG := DG ∪ {Dg'_i};
```

F. CkTailv2 Step 4: IOLTS Generation

This last step, implemented by Algorithm 5, generates one IOLTS for every component in C. The algorithm starts by

Algorithm 5: IOLTS Generation

```
   input  : Traces(SUL)
   output : IOLTSs L_{c_1} ... L_{c_k}
1  T := {};
2  foreach σ = a_1(α_1) ... a_k(α_k) ∈ Traces(SUL)) do
3      σ' := ϵ;
4      foreach a_i(α_i) ⪯ σ do
5          σ' := σ'.!a_i(α_i ∪ {id := from(a_i(α_i))} \ {time :=
               t, session := s});
6          if isReq(a_i(α_i)) ∨ isResp(a_i(α_i))) then
7              σ' := σ'.?a_i(α_i ∪ {id := to(a_i(α_i))} \ {time :=
                   t, session := s});
8      foreach c ∈ C do
9          T_c := T_c ∪ {σ' \ {a(α) ∈ σ' | (id := c) ∉ α}
10 foreach T_c with c ∈ C do
11     Generate the IOLTS L_c from T_c;
12     Merge the equivalent states of L_c with kTail(k = 2, L_c);
```

transforming the traces to integrate the notions of input and output. Given a trace $a_1(\alpha_1)\dots a_k(\alpha_k)$, every action is doubled by separating the component source and destination. The source and the destination are identified by a new assignment on the parameter id added to each action. Besides, the timestamps and session identifiers are removed from the assignments to improve the model generalisation. For a communication action $a_i(\alpha_i)$, this step produces a new trace σ' composed of the output $!a_i(\alpha_{i1})$ sent by the source of the message, followed by the input $?a_i(\alpha_{i2})$ received by the destination (lines 5-7). Non-communication actions are marked as outputs. Then, this new trace σ' is segmented into sub-sequences, each capturing the behaviours of one component only (lines 8, 9). The trace set T_c gathers the traces of the component c.

Every trace set T_c is now lifted to the level of IOLTS. A trace $t = a_1(\alpha_1)\dots a_k(\alpha_k) \in T_c$ is transformed into the path $q0 \xrightarrow{a_1(\alpha_1)} q_1 \dots q_{k-1} \xrightarrow{(a_k(\alpha_k)} q0$ such that the states $q_1 \dots q_{k-1}$ are new states. These paths are joined on the state $q0$ to build the IOLTS L_c:

Definition 4 (IOLTS generation) *Let $T_c = \{t_1, \dots, t_n\}$ be a trace set. $L_c = \langle Q, q0, \Sigma, \rightarrow \rangle$ is the IOLTS derived from T_c where:*

- *$q0$ is the initial state.*
- *$Q, \Sigma, \rightarrow$ are defined by the following rule:*

$$\frac{t_i = a_1(\alpha_1)\dots a_k(\alpha_k)}{q0 \xrightarrow{a_1(\alpha_1)} q_{i1}\dots q_{ik-1} \xrightarrow{a_k(\alpha_k)} q0}$$

Finally, Algorithm 5 applies the kTail algorithm to generalise and reduce the IOLTSs by merging the equivalent states having the same k-future. We use $k = 2$ as recommended in [4, 41].

V. Empirical Evaluation

The experiments presented in this section aim to evaluate the capabilities of our algorithms to build models in terms of precision and performance, compared to the approaches allowing to learn models of communicating systems. Prior to this work, we evaluated CkTailv1 along with the tools CSight [9], Assess [26], and the tool suite proposed in [19] based upon the tool kbehavior, which we denote LFkbehavior. Our experimental results, given in [2], showed that Assess requires assumptions that are strongly different than those required by the other tools. The main difference for Assess lies in the fact that the communications among components are assumed hidden (not available in event logs). Assess tries to detect implicit calls of components instead, and completes models with synchronisation actions to express them. When this assumption does not hold, i.e., when we feed Assess with event logs including communication messages, we showed that it builds high imprecise models. Consequently, for this new evaluation, we have chosen to conduct several experimentations on CSight, LFkbehavior, CkTailv1 and CkTailv2 (source code and explanations available in [20]). As our approach uses two distinct trace extraction algorithms, we have chosen to differentiate them with the notations CkTailv2-w/oS (Algorithm 1 without session identifier) and CkTailv2-w/S (Algorithm 3 with session identifiers).

This evaluation aims at investigating the capabilities of our algorithms through the following four questions:

- RQ1: can CkTailv2 infer models that capture correct behaviours of SUL? This question studies the capability of CkTailv2 to build models that accept valid traces of the system compared to CSight, CkTailv1 and LFkbehavior. The valid traces correspond to traces extracted from event logs but not used for the model generation;
- RQ2: do the models inferred by CkTailv2 reject abnormal behaviours? RQ2 studies the capability of CkTailv2 to generate models that reject invalid traces, compared to CSight, CkTailv1 and LFkbehavior. Invalid traces express abnormal behaviours of the system;
- RQ3: is CkTailv2 able to detect accurate dependencies among components? RQ3 investigates the recall and precision of CkTailv2 to detect component dependencies. Recall is here the percentage of the real dependencies that are detected, and precision is the percentage of detected dependencies that are correct;
- RQ4: what is the performance of CkTailv2 to infer models compared to the other tools? How does CkTailv2 scale with the size of the event log?

A. Empirical Setup

To generate models, the considered tools impose different assumptions, which we examined before our experiments to avoid any bias. We ran LFkbehavior with the strategy that segments event logs w.r.t. component identifier, as this is the only one that can be applied with communicating systems to build one model per component. CSight does not take event log as input but trace sets such that every component is associated to its own trace set. CkTailv1 is more restrictive on the event log content than LFkbehavior and CkTailv2. For CkTailv1, an event log must be exclusively composed of communication events and a request must be associated to one response only.

As a consequence, we have taken into consideration all these differences through experiments conducted on several

configurations. We firstly assembled and configured 6 communicating systems from a set of 7 commercial devices (3 sensors, 2 gateways, 2 actuators). Each of these systems contains at least one gateway using the home automation system Domoticz [1], connected to at least two sensors and one actuator. The behaviours of the gateway(s) after the receipt of data from the sensors differ in each configuration. We monitored these systems and collected event logs of about 2200 events. We denote them $Conf1$ to $Conf6$. We also considered 2 other systems made up of other components to avoid giving conclusions on similar systems. The first one has 8 sensors (4 are commercial devices and the others are based upon the open source framework EspEasy [2]) that periodically send data to a Cloud server. The second one corresponds to an IP security camera, which is interconnected to NTP, SMTP and FTP servers. The corresponding event logs are denoted $Conf7$ and $Conf8$ and respectively include 2206 and 1310 events. All these event logs do not include session identifiers. Hence, we manually modified them to compare our algorithms CkTailv2-w/oS and CkTailv2-w/S. The modifications consisted in adding a session identifier in every action with regard to the functioning of the systems. We denote these new event logs $Conf9$ to $Conf16$.

All the tools except CSight take event logs as input. We experimented CSight after having manually segmented $Conf1$ to $Conf8$ into trace sets, but we were unable to get any result after 5 hours of computation, which was our limit for each experiment. We observed that the first steps of CSight were achieved, but these were always followed by time-outs. The last steps of Csight call a model-checker to refine models with invariants, and we suspect that the model-checker was unable to check invariant satisfiability on large trace sets. Therefore, to compare CSight with the other tools, we took back two trace sets given with the CSight implementation. The first one, denoted Tcp contains 8 traces (46 events) collected from two components exchanging TCP messages. The second trace set denoted $AltBit$ contains 15 traces (246 events) expressing message exchanges between two components over the Alternating Bit Protocol, which belongs to the family of reliable transport protocols.

In summary, we considered 18 configurations. $Conf1$, 3, 5, 8, Tcp and $AltBit$ are event logs that meet the requirements of all the tools, and are particularly interesting for comparing CSight, CkTailV1, LFkbehavior and CkTailv2-w/oS. $Conf2$, 4, 6, 7 are more general event logs (composed of requests associated to multiple responses and of non-communication events) and are used to confront CkTail- v2-w/oS with LFkbehavior. Finally, $Conf9$ to 16 allow to compare our algorithms CkTailv2-w/oS and CkTailv2-w/S.

Furthermore, CkTailv1 and CkTailv2 use the procedure $ontime$ to check whether an action belongs to a current session with regard to the session duration. The same procedure, which is given in Section IV-C, was used for both tools.

[1]https://www.domoticz.com/

[2]https://www.letscontrolit.com/

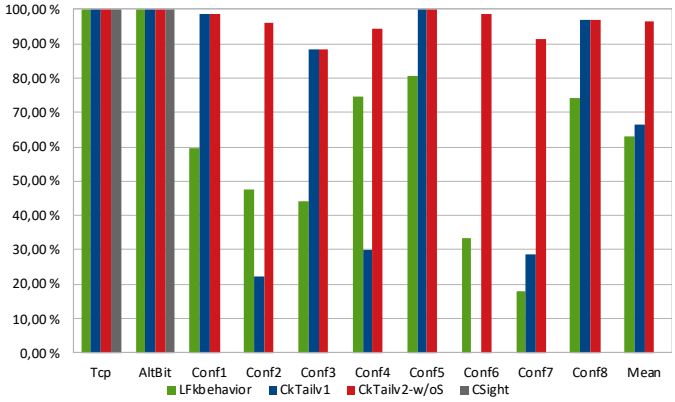


Fig. 8. Percentage of valid traces accepted by the models with the configurations Tcp, $AltBit$, $Conf1$ to 8

B. RQ1: can CkTailv2 infer models that capture correct behaviours of SUL*?*

To answer RQ1, we measured the rate of valid traces accepted by all the behavioural models generated from the 18 configurations. Given a valid trace σ and an IOLTS L, IOLTS acceptance means here that $\sigma \in Traces(L)$. To get valid traces, we chose to follow a Hold Out method, which partitioned each event log in one training log for the model generation and one testing log for the extraction of valid traces. We manually segmented event logs into two parts with an approximative ratio of 80% and 20%, taking care not to separate actions that belong to the same session to avoid the generation of incorrect models.

Afterwards, still to avoid any bias, we extracted valid trace sets from the testing logs. This trace extraction was automatically performed for the event logs including session ids. But for the other event logs, as there is no information allowing to recognise valid traces, we manually extracted them by leveraging our knowledge of the case study functioning.

We obtained around 35 to 200 valid traces for $Conf1$ to 16. For the configurations Tcp and $AltBit$ we respectively used 75% of the traces to generate models, the remaining being used as valid traces.

a) Results: The percentages of valid traces accepted by the models generated by each tool are illustrated in the bar-diagrams of Figures 8 and 9. With the configurations $Conf1$ to 8, the models that accept the most of valid traces are always those generated by CkTailv2-w/oS. In our experiments, these models accept an average of 96.43% of valid traces. The models given by CkTailv1 and LFkbehavior provide close results with 66.47% and 63.23%. If we focus on the results given by CkTailv2-w/oS and CkTailV1, we have the same rate of valid traces accepted by the models with $Conf1$, 3, 5 and 8. These similarities come from the fact that these configurations meet the assumptions of both tools. The trace segmentation along with the model generation are hence performed in a similar manner. As expected, with the other configurations, we observe that CkTailv1 produced less correct

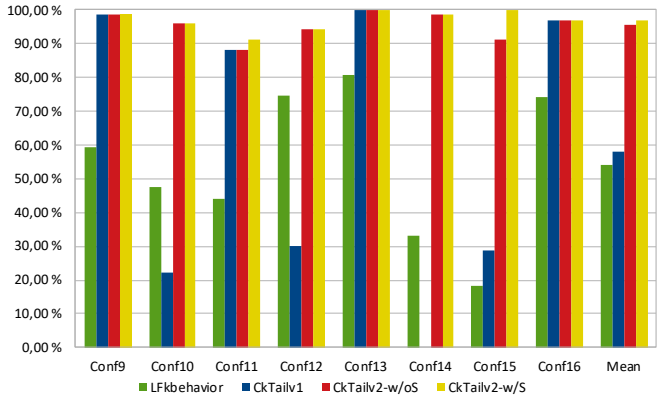


Fig. 9. Percentage of valid traces accepted by the models with the configurations $Conf9$ to 16

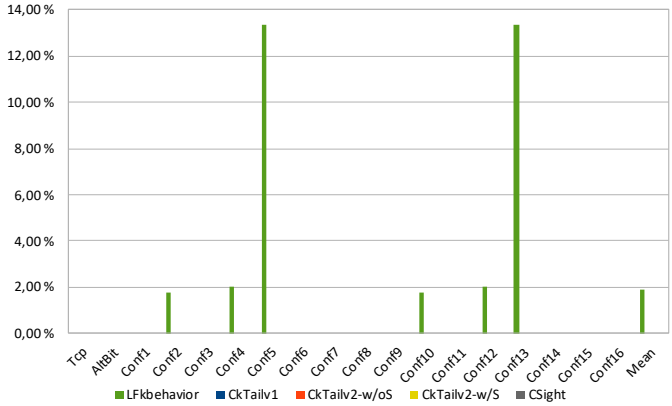


Fig. 10. Percentage of invalid traces accepted by the models for each configuration

models as it eliminated some actions during the trace extraction. LFkbehavior outperforms CkTailv1 with $Conf2$, 4 and 6 for the same reason.

Figure 8 also shows that the models generated from the configurations Tcp and $AltBit$ accept all the valid traces, whatever the approach used. These results tend to reveal that the sizes of the event logs used with these configurations are not large enough to make distinctions among the approaches. Therefore, we prefer to not give any conclusion here. As stated earlier, we were unable to apply CSight on larger trace sets.

Figure 9 shows that when the event logs include sessions identifiers, LFkbehavior and CkTailv1 infer models accepting the same ratio of valid traces. The interesting observation is that CkTailv2-w/oS and CkTailv2-w/S provide close results, i.e., the models given by CkTailv2-w/oS accept slightly less valid traces only. We recall that CkTailv2-w/S extracts traces from event logs by means of session identifiers (the trace extraction is always correct) whereas CkTailv2-w/oS tries to detect sessions for extracting traces. Hence, Figure 9 tends to show that the trace extraction algorithm of CkTailv2-w/oS (Algorithm 1) is very effective.

C. RQ2: do the models inferred by CkTailv2 reject abnormal behaviours?

This research question targets the capability of our algorithms to infer models that reject incorrect behaviours of the system. Incorrect behaviours are expressed by means of invalid traces, which are here derived from valid traces by injecting one of the following errors: repetition of actions (random addition of 2 to 6 actions), inversion of a request with its associated response(s), permutation of one request in a sequence of nested requests, and suppression of one response when several responses associated to the same request are found.

We generated 16 sets having 43 to 100 invalid traces for each configuration $Conf1$ to 16, and two sets of 20 invalid traces for Tcp and for $AltBit$. Then, we measured the proportions of invalid traces accepted by the range of models inferred from the same configurations and training sets used for RQ1.

a) Results: The bar-diagram of Figure 10 shows the proportions of invalid traces accepted by the models given by each tool in each configuration. This figure reveals that all the tools performed well in the sense that the inferred models reject most of the incorrect behaviours. CkTailv1, CkTailv2 and CSight outperform LFkbehavior with some configurations though. For instance, LFkbehavior produced models that accept 13.3% of invalid traces with $Conf5$.

As previously, it remains difficult to compare CSight and CkTailv2 because only two configurations Tcp and $AltBit$ are considered in this evaluation. As CSight uses invariant satisfiability to increase the model precision and not CkTailv2, we believe that CSight should return more precise models, but only with small trace sets.

The results given with RQ1 and RQ2 tend to indicate that the models produced by CkTailv2 offer the best precision: not only they accept the highest ratio of valid traces, but also reject all the invalid ones (as CSight).

D. RQ3: can CkTailv2 detect accurate dependencies among components?

This research question investigates the capability of our algorithms to find component dependencies during the event log analysis. Among the range of tools considered in this evaluation, only CkTailv1, CkTailv2-w/oS and CkTailv2-w/S are able to infer dependency graphs, but CkTailv1 and CkTailv2-w/oS use the same dependency detection. As a consequence, we chose to study RQ3 by comparing the DAGs returned by CkTailv2-w/oS and CkTailv2-w/S to the real dependency graphs we manually built from the dependency schemes that we devised for $Conf1$ to 8, Tcp and $AltBit$. We evaluated the recall and precision of both algorithms. A good component dependency detection is characterised by a high recall and high precision, where high recall also relates to a low false negative rate and high precision relates to low false positive rate.

a) Results: Table III shows the number of real component dependencies for each configuration and the bar-diagram of Figure 11 depicts the recalls and precisions achieved by

TABLE III
REAL DEPENDENCIES FOR EACH CONFIGURATION

$Conf1$	$Conf2$	$Conf3$	$Conf4$	$Conf5$	$Conf6$	$Conf7$	$Conf8$	Tcp	$AltBit$
8	8	9	6	8	7	4	10	2	2

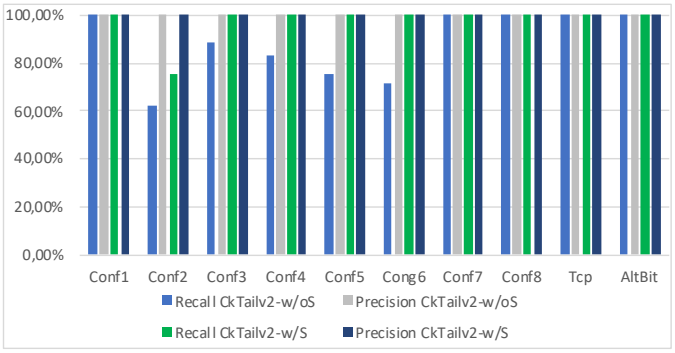


Fig. 11. Recall and precision of CkTailv2 to detect component dependencies. Recall is the percentage of the real dependencies that are detected; precision is the percentage of detected dependencies that are correct

CkTail- v2-w/oS and CkTailv2-w/S. On average, CkTailv2-w/oS detected 88% of the real dependencies and CkTailv2-w/S 97.5%. No wrong component dependency is returned by both algorithms. After inspection, we observed that the undiscovered dependencies correspond to some data dependencies that can be observed among several chains of messages sharing the same data addressed to several components at the same time. We have chosen in Definition 3 to not consider them to avoid returning false dependencies. This case of having chains of messages sharing the same data addressed to several components is more frequent with CkTailv2-w/oS as it detects data dependencies on the action sequence S, while CkTailv2-w/S does it on traces, which are smaller sequences. As a consequence, the recall of CkTailv2-w/oS is lower than the one of CkTailv2-w/S.

E. RQ4: what is the performance of CkTailv2 to infer models compared to the other tools? How does CkTailv2 scale with the size of the event log?

a) Procedure: To answer RQ4, we firstly studied how the tools scale with the size of the event logs. We collected 40 event logs from $Conf3$ by varying the number of events between 500 to 10000 events. Then, we measured execution times to produce models. As CSight did not complete on $Conf3$, we considered LFkbehavior, CkTailv1, CkTailv2-w/oS and CkTailv2-w/S. Besides, ss CkTailv2-w/S has two modes, i.e., trace extraction with session identifiers provided by the user, and extract of identifiers when these are not provided, we applied both modes on $Conf3$. For readability, this is denoted as CkTailv2-w/S and CkTailv2-w/US. To include CSight in our evaluation, we measured the execution times of all the tools on Tcp and $AltBit$. Experiments were carried out on a computer with 1 Intel® CPU i5-6500 @ 3.2GHz and 32GB RAM.

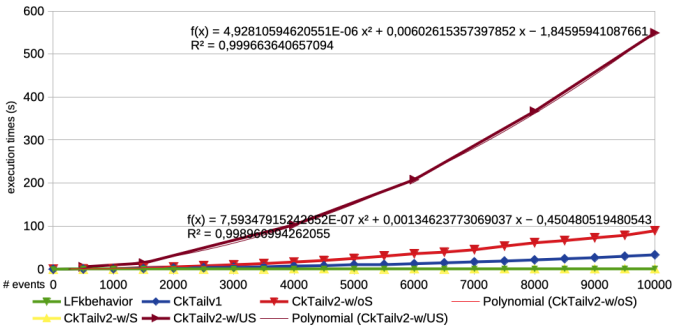


Fig. 12. Execution times vs. number of events

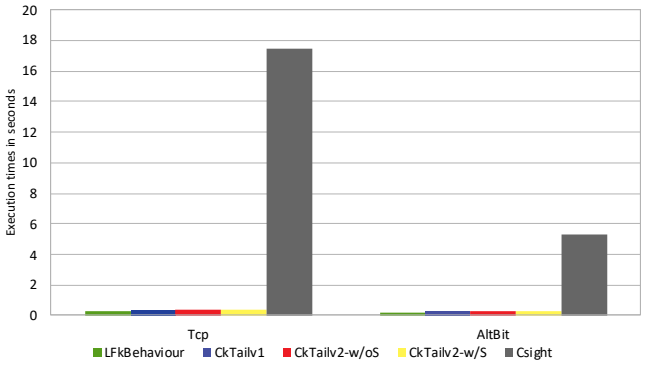


Fig. 13. Execution times of the tools with the configurations Tcp and $AltBit$

b) Results: Figure 12 depicts the execution times in seconds of the tools w.r.t. the event log sizes. CkTailv2-w/S offers the best performance as it produces models in less than 1 second. These results are not surprising as the algorithm splits event logs quickly thanks to the known session identifiers. LFkbehavior offers close results as it never took more than 2 seconds to produce models. On the other hand, CkTailv1 and CkTailv2-w/oS required less than 33s and 89s, respectively. The curve for CkTailv2-w/oS follows a quadratic regression. The difference between CkTailv1 and CkTailv2-w/oS comes from the fact that CkTailv2-w/oS uses two set of pending requests to check if the constraints C1-C5 hold while CkTailv1 needs one set only. The last curve shows execution times with CkTailv2-w/US. In this case, the curve also follows a quadratic regression but reveals that our tool does not scale well. Most of the execution times are consumed by the analysis of the event logs to recover session identifiers. These are indeed retrieved by testing whether combinations of parameter assignments identify execution traces w.r.t. the satisfiability of session patterns and the evaluation of trace quality metrics.

The bar-diagram of Figure 13 illustrates the execution times of all the tools on the configurations Tcp and $AltBit$. These results tend to show that CSight is significantly slower than the other tools, it is around 30 times slower than CkTail- v2-w/oS. Besides, as stated earlier, CSight were unable to return models after 5 hours with $Conf1$ to 8.

These experiments show that CkTailv2 can be used in practice to infer models of communicating systems even with large event logs, but it suffers from insufficient scalability, on account of its feature of detecting sessions for extracting traces.

F. Threat to Validity

Some threats to validity can be identified in our evaluation. The first factor, which may threaten the external validity of our results, applies to the case studies used in the experimentations. Most of them indeed are IoT systems using the HTTP protocol. We also considered two other event logs collected from components exchanging messages by means of the TCP and Alternating bit protocols. But many communicating systems rely on other kinds of protocols, from which it may be more difficult to identify senders, receivers, requests or responses. Hence, our results cannot be generalised to all communicating systems. This is why we deliberately avoid drawing any general conclusion. We chose to mainly concentrate our experimentations on IoT systems that we devised to be able to appraise the capability of CkTailv2 of inferring correct dependency graphs. This threat is somewhat mitigated by the fact that our results can be easily generalised to communicating systems based upon the HTTP protocol, and that the latter is used by numerous communicating systems.

The generalisation of our approach is also restricted by the requirements A1-A3. The event logs have to include timestamps given by a global clock and must be formatted by means of regular expressions so that the event types can be identified. Although we have observed that this task is not too difficult to carry out on HTTP messages, it is manifest that this is not generalisable to any kind of protocols, especially those encrypting some parts of the message contents. We need to investigate how these requirements could be relaxed in future work.

There are also some threats to internal validity. Firstly, like all the other passive model learning approaches, the larger the event log, the more complete and precise the models will be. Furthermore, our approach uses one parameter denoted T, in the procedure $ontime$, to limit the session duration. We set this parameter to 1 or 2 seconds in our experiments as the session durations was lower than these values in our case studies. Changing this parameter impacts the precision of the models though. We assume that the user has some knowledge about SUL and that he/she can set this parameter correctly. Otherwise, we suggest to generate several models while modifying this parameter. We evaluated the precisions of the models generated from $Conf2$ with T taking values between 0 and 150 seconds. Figure 14 illustrates the ratios of valid and invalid traces accepted by the inferred models. The ratio of invalid traces remain unchanged. But, the ratio of valid traces evolves with T. Although the figure does not allow to directly deduce the best parameter value as several ones are possible, it helps avoid choosing the bad ones.

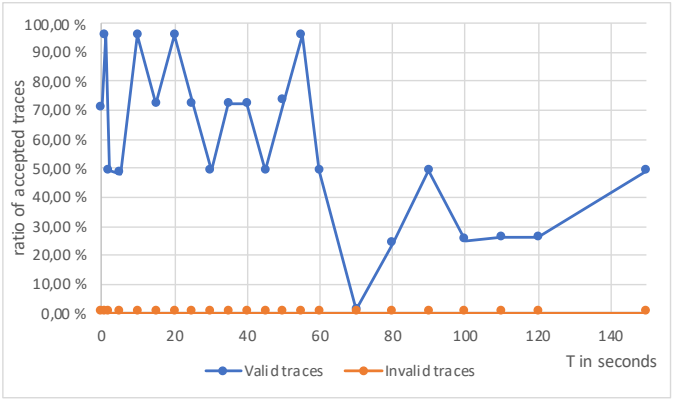


Fig. 14. Impact of the session duration on the model precision

VI. Conclusion

This paper has proposed the design and evaluation of a tool called CkTailv2, which is specialised into the learning of behavioural models along with dependency graphs from event logs, themselves collected from of communicating systems. Compared to other model learning algorithms, CkTailv2 increases the precision of the generated models by integrating an algorithm that better recognises sessions in event logs with respect to constraints related to the request-response pattern, the recognition of nested requests, time delays and component dependency. Besides, when sessions are explicitly identified in event logs, CkTailv2 provides another algorithm to quicker generate models.

CkTailv2 is simple to use. A user only has to give an event log and a set of regular expressions as inputs to produce one IOLTS and one DAG per component of the communicating system. These models are stored in DOT files and varied tools can process them to graphically represent how the communicating system behaves and is structured. These models may be then used to detect defects or security vulnerabilities. Besides, our evaluation showed that CkTailv2 is effective, as it provides precise models, and that it can be used in practice on large event logs.

Nevertheless, several aspects need to be investigated and improved in the future. We firstly plan to evaluate CkTailv2 on further kinds of systems to confirm our experimental results. The latter show that CkTailv2 does not scale well with the size of the event logs. We believe that the performance can be improved by devising parallel algorithms. But another way is to get rid of some requirements, such as the need to have events that encode senders and receivers. We believe that an additional event log analysis step could perform this task automatically.

Another direction of future work is to make use of these models to assist developers in the analysis and test of communicating systems. More precisely, we intend to propose an approach combining this model learning technique with the generation of mocks, i.e., fake components that simulate real components and that behave in a predefined way. These mock components could make test development easier by replacing

complex dependencies (e.g., infrastructure or environment related dependencies [42]). Besides, mock components could increase test efficiency by replacing slow-to-access components. We finally believe that the models produced by CkTailv2 could be analysed to automatically generate executable mock components.

VII. Acknowledgement

Research supported by the French Project VASOC (Auvergne-Rhône-Alpes Region) https://vasoc.limos.fr/

References

[1] S. Salva, "Reverse Engineering Models of Concurrent Communicating Systems From Event Logs," in *Sixteenth International Conference on Software Engineering Advances ICSEA 2021*, Barcelona, online, Spain, Oct. 2021, pp. 37–42. [Online]. Available: https://hal.uca.fr/hal-03444549

[2] S. Salva and E. Blot, "Cktail: Model learning of communicating systems," in *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2020, Prague, Czech Republic, May 5-6, 2020*, R. Ali, H. Kaindl, and L. A. Maciaszek, Eds. SCITEPRESS, 2020, pp. 27–38. [Online]. Available: https://doi.org/10.5220/0009327400270038

[3] A. Biermann and J. Feldman, "On the synthesis of finite-state machines from samples of their behavior," *Computers, IEEE Transactions on*, vol. C-21, no. 6, pp. 592–597, June 1972.

[4] D. Lorenzoli, L. Mariani, and M. Pezzè, "Automatic generation of software behavioral models," in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE'08. New York, NY, USA: ACM, 2008, pp. 501–510.

[5] F. Pastore, D. Micucci, and L. Mariani, "Timed k-tail: Automatic inference of timed automata," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, March 2017, pp. 401–411.

[6] L. Mariani and M. Pezze, "Dynamic detection of cots component incompatibility," *IEEE Software*, vol. 24, no. 5, pp. 76–85, 2007.

[7] I. Beschastnikh, Y. Brun, S. Schneider, M. Sloan, and M. D. Ernst, "Leveraging existing instrumentation to automatically infer invariant-constrained models," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE '11. New York, NY, USA: ACM, 2011, pp. 267–277.

[8] T. Ohmann, M. Herzberg, S. Fiss, A. Halbert, M. Palyart, I. Beschastnikh, and Y. Brun, "Behavioral resource-aware model inference," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '14. New York, NY, USA: ACM, 2014, pp. 19–30.

[9] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy, "Inferring models of concurrent systems from logs of their behavior with csight," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 468–479. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568246

[10] L. Mariani, M. Pezzè, and M. Santoro, "Gk-tail+ an efficient approach to learn software models," *IEEE Transactions on Software Engineering*, vol. 43, no. 8, pp. 715–738, Aug 2017.

[11] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87–106, 1987.

[12] P. Dupont, "Incremental regular inference," in *Proceedings of the Third ICGI-96*. Springer, 1996, pp. 222–237.

[13] H. Raffelt, B. Steffen, and T. Berg, "Learnlib: A library for automata learning and experimentation," in *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems*, ser. FMICS '05. New York, NY, USA: ACM, 2005, pp. 62–71.

[14] R. Alur, P. Černý, P. Madhusudan, and W. Nam, "Synthesis of interface specifications for java classes," *SIGPLAN Not.*, vol. 40, no. 1, pp. 98–109, Jan. 2005.

[15] T. Berg, B. Jonsson, and H. Raffelt, "Regular inference for state machines with parameters," in *Fundamental Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, L. Baresi and R. Heckel, Eds. Springer Berlin Heidelberg, 2006, vol. 3922, pp. 107–121.

[16] F. Howar, B. Steffen, B. Jonsson, and S. Cassel, "Inferring canonical register automata," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, V. Kuncak and A. Rybalchenko, Eds. Springer Berlin Heidelberg, 2012, vol. 7148, pp. 251–266.

[17] K. Hossen, R. Groz, C. Oriat, and J. Richier, "Automatic model inference of web applications for security testing," in *Seventh IEEE International Conference on Software Testing, Verification and Validation, ICST 2014 Workshops Proceedings, March 31 - April 4, 2014, Cleveland, Ohio, USA*, 2014, pp. 22–23.

[18] A. Petrenko and F. Avellaneda, "Learning communicating state machines," in *Tests and Proofs - 13th International Conference, TAP 2019, Held as Part of the Third World Congress on Formal Methods 2019, Porto, Portugal, October 9-11, 2019, Proceedings*, ser. Lecture Notes in Computer Science, D. Beyer and C. Keller, Eds., vol. 11823. Springer, 2019, pp. 112–128. [Online]. Available: https://doi.org/10.1007/978-3-030-31157-5

[19] L. Mariani and F. Pastore, "Automated identification of failure causes in system logs," in *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*, Nov 2008, pp. 117–126.

[20] E. Blot and S. Salva, "The cktailv2 tool," 2020. [Online]. Available: https://github.com/sasa27/CkTailv2

[21] G. Ammons, R. Bodík, and J. R. Larus, "Mining specifications," *SIGPLAN Not.*, vol. 37, no. 1, pp. 4–16, Jan. 2002.

[22] B. K. Aichernig and M. Tappler, "Learning from faults: Mutation testing in active automata learning - mutation testing in active automata learning," in *NASA Formal Methods - 9th International Symposium, NFM 2017, Moffett Field, CA, USA, May 16-18, 2017, Proceedings*, 2017, pp. 19–34.

[23] R. Groz, K. Li, A. Petrenko, and M. Shahbaz, "Modular system verification by inference, testing and reachability analysis," in *Testing of Software and Communicating Systems*, K. Suzuki, T. Higashino, A. Ulrich, and T. Hasegawa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 216–233.

[24] M. Tappler, B. K. Aichernig, and R. Bloem, "Model-based testing iot communication via active automata learning," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, March 2017, pp. 276–287.

[25] S. Salva and E. Blot, "Model generation of component-based systems," *Software Quality Journal*, vol. 28, no. 2, pp. 789–819, June 2020.

[26] ——, "Reverse engineering behavioural models of iot devices," in *31st International Conference on Software Engineering & Knowledge Engineering (SEKE)*, Lisbon, Portugal, July 2019. [Online]. Available: https://hal-clermont-univ.archives-ouvertes.fr/hal-02134046

[27] ETSI, "Methods for testing & specification; risk-based security assessment and testing methodologies, european telecommunications standards institute, technical report, https://www.etsi.org/deliver/etsi_eg/203200_203299/203251/01.01.01_50/eg_203251v010101m.pdf," 2015.

[28] R. Dssouli, K. Karoui, A. Petrenko, and O. Rafiq, "Towards testable communication software," in *Protocol Test Systems VIII: Proceedings of the IFIP WG6.1 TC6 Eighth International Workshop on Protocol Test Systems, September 1995*, A. Cavalli and S. Budkowski, Eds. Boston, MA: Springer US, 1996, pp. 237–251.

[29] S. Salva, H. Fouchal, and S. Bloch, "Metrics for timed systems testing," in *Procedings of the 4th International Conference on Principles of Distributed Systems, OPODIS 2000, Paris, France, December 20-22, 2000*, 2000, pp. 177–200.

[30] E. Blot and S. Salva, "Testability measurments on inffered models," 2020. [Online]. Available: https://github.com/Elblot/testability

[31] L. Gutiérrez-Madroñal, I. Medina-Bulo, and J. Domínguez-Jiménez, "Iot–teg: Test event generator system," *Journal of Systems and Software*, vol. 137, pp. 784–803, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121217301280

[32] A. Ahmad, F. Bouquet, E. Fourneret, F. Le Gall, and B. Legeard, "Model-based testing as a service for iot platforms," in *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2016, pp. 727–742.

[33] S. Salva and E. Blot, "Verifying the application of security measures in iot software systems with model learning," in *Proceedings of the 15th 15th International Conference on Software Technologies, ICSOFT 2020, Paris, France, july, 2020*, 2020, pp. 1–12.

[34] Y. Falcone, M. Jaber, T.-H. Nguyen, M. Bozga, and S. Bensalem, "Runtime Verification of Component-Based Systems," in *SEFM 2011 - Proceedings of the 9th International Conference on Software Engineering and Formal Methods*, ser. Lecture Notes in Computer Science (LNCS), G. Barthe, A. Pardo, and G. Schneider, Eds., vol. 7041. Montevideo, Uruguay: Springer, Nov. 2011, pp. 204–220. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00642969

[35] M. van der Bijl, A. Rensink, and J. Tretmans, "Compositional testing with ioco," in *Formal Approaches to Software Testing*, A. Petrenko and A. Ulrich, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 86–100.

[36] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," *2009 Ninth IEEE International Conference on Data Mining*, pp. 149–158, 2009.

[37] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 1921–1936, Nov 2012.

[38] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 1–7.

[39] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, and R. Sasnauskas, "A search-based approach for accurate identification of log message formats," in *Proceedings of the 26th Conference on Program Comprehension*, ser. ICPC '18. New York, NY, USA: ACM, 2018, pp. 167–177. [Online]. Available: http://doi.acm.org/10.1145/3196321.3196340

[40] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," *CoRR*, vol. abs/1811.03509, 2018. [Online]. Available: http://arxiv.org/abs/1811.03509

[41] D. Lo, L. Mariani, and M. Santoro, "Learning extended fsa from software: An empirical assessment," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2063 – 2076, 2012, selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212001008

[42] D. Spadini, M. Aniche, M. Bruntink, and A. Bacchelli, "Mock objects for testing java systems," *Empirical Softw. Eng.*, vol. 24, no. 3, p. 1461–1498, Jun. 2019. [Online]. Available: https://doi.org/10.1007/s10664-018-9663-0

Toward Scalable Collaborative Metaprogramming: A Case Study to Integrate Two Metaprogramming Environments

Herwig Mannaert
University of Antwerp
Antwerp, Belgium
Email: herwig.mannaert@uantwerp.be

Chris McGroarty
U.S. Army Combat Capabilities Development
Command Soldier Center (CCDC SC)
Orlando, Florida, USA
Email: christopher.j.mcgroarty.civ@mail.mil

Scott Gallant
Effective Applications Corporation
Orlando, Florida, USA
Email: Scott@EffectiveApplications.com

Koen De Cock
NSX BV
Niel, Belgium
Email: koen@nsx.normalizedsystems.org

Jim Gallogly
Cole Engineering Services Inc.
Orlando, Florida, USA
Email: james.gallogly@cesicorp.com

Anup Raval and Keith Snively
Dynamic Animation Systems
Fairfax, Virginia, USA
Email: araval,ksnively@d-a-s.com

***Abstract*—The automated generation of source code, often referred to as metaprogramming, has been pursued for decades in computer programming. Though many such metaprogramming environments have been proposed and implemented, scalable collaboration within and between such environments remains challenging. It has been argued in previous work that a meta-circular metaprogramming architecture, where the the metaprogramming code (re)generates itself, enables a more scalable collaboration and easier integration. In this contribution, an explorative case study is performed to integrate this meta-circular architecture with another metaprogramming environment. Based on a detailed description of the architectures of both metaprogramming environments, the various technical aspects and issues concerning this integration are analyzed. Some preliminary results from applying this approach in practice are presented and discussed.**

***Index Terms*—Evolvability; Normalized Systems; Simulation Models; Automated programming; Case Study**

I. Introduction

This paper extends a previous paper which was originally presented at the Fifteenth International Conference on Software Engineering Advances (ICSEA) 2020 [1].

The automated generation of source code, often referred to as automatic programming or metaprogramming, has been pursued for decades in computer programming. Though the increase of programming productivity has always been an important goal of automatic programming, its value is of course not limited to development productivity. Various disciplines like systems engineering, modeling, simulation, and business process design could reap significant benefits from metaprogramming techniques.

While many implementations of such automatic programming or metaprogramming exist, many people believe that automatic programming has yet to reach its full potential [2][3]. Moreover, where large-scale collaboration in a single metaprogramming environment is not straightforward, realizing such a scalable collaboration between different metaprogramming environments is definitely challenging.

In our previous work [4] [5], we have presented a meta-circular implementation of a metaprogramming environment, and have argued that this architecture enables a scalable collaboration between various metaprogramming projects. In this contribution, we perform an explorative case study to perform a first integration with another metaprogramming environment. To remain generic, the two metaprogramming environments are aimed at generative programming for completely different types of software systems, and based on totally different meta-models. At the same time, they are well suited for this study, as they both pursue a more horizontal integration architecture. The case study aims to serve as an architectural pathfinder for such integrations, and to identify remaining issues that hamper the scalability of the approach.

The remainder of this paper is structured as follows. In Section II, we briefly present some aspects and terminology with regard to metaprogramming, and argue the relevance of two related concepts: meta-circularity and systems integration. Based on this concept of systems integration, we argue for more horizontal integration architectures to enable scalable collaboration. The next two sections present the architecture and meta-model of both metaprogramming environments whose integration is explored in this contribution. Section III discusses the Normalized Systems metaprogramming environment and refers rather extensively to previous work. Section IV offers a detailed architectural description of the generative programming environment for simulation models. Based on these architectures, Section V elaborates on the integration of these metaprogramming environments, detailing the various

technical aspects, the achieved progress, and the remaining issues. Finally, we present some conclusions in Section VI.

II. Metaprogramming and Systems Integration

In this section, we give an overview of the main concepts and terminology regarding metaprogramming, and discuss the related concept of meta-circularity. Based on the basic charasteristics of metaprogramming, we propose to leverage the technique of systems integration to pursue collaborative and scalable metaprogramming. We also argue that the two selected metaprogramming environments are well suited for a representative case study.

A. Metaprogramming Concepts and Meta-Circularity

The automatic generation of source code is probably as old as software programming itself, and is in general referred to by various names. *Automatic programming*, stresses the act of automatically generating source code from a model or template, and has been called ”a euphemism for programming in a higher-level language than was then available to the programmer” by David Parnas [6]. *Generative programming*, ”to manufacture software components in an automated way” [7], emphasizes the manufacturing aspect and the similarity to production and the industrial revolution. *Metaprogramming*, sometimes described as a programming technique in which ”computer programs have the ability to treat other programs as their data” [8], stresses the fact that this is an activity situated at the meta-level, i.e., writing software programs that write software programs.

Academic papers on metaprogramming based on intermediate representations or *Domain Specific Languages (DSLs)*, e.g., [9], focus in general on a specific implementation. Also related to metaprogramming are software development methodologies such as *Model-Driven Engineering (MDE)* and *Model-Driven Architecture (MDA)*, requiring and/or implying the availability of tools for the automatic generation of source code. Today, these model-driven code generation tools are often referred to as *Low-Code Development Platforms (LCDP)* or *No-Code Development Platforms (NCDP)*, i.e., software that enables developers to create application software through configuration instead of traditional programming. This field is still evolving and facing criticisms, as some question whether these platforms are suitable for large-scale and mission-critical enterprise applications [2], while others even question whether these platforms actually make development cheaper or easier [3]. Moreover, defining an intermediate representation or reusing DSLs is still a subject of research today. We mention the contributions of Wortmann [10], presenting a novel conceptual model for the systematic reuse of DSLs, and Gusarov et al. [11], proposing an intermediate representation to be used for code generation.

Concepts somewhat related to metaprogramming are homoiconicity and meta-circularity. Both concepts refer to some kind of circular behavior, and are also aimed at the increase of the abstraction level, and thereby the productivity of computer

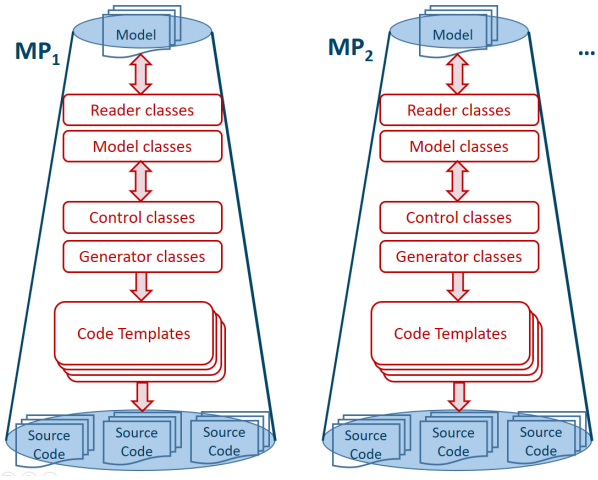


Fig. 1. Representation of the duplication of metaprogramming silos.

programming. Homoiconicity is specifically associated with a language that can be manipulated as data using that language, and traces back to the design of the language TRAC [12], and to similar concepts in an earlier paper from McIlroy [13]. Meta-circularity, first coined by Reynolds describing his meta-circular interpreter [14], expresses the fact that there is a connection or feedback loop between the meta-level, the internal model of the language, and the actual models or code expressed in the language. Such circular properties have the potential to be highly beneficial for metaprogramming through a reduction of complexity for the metaprogrammers. Indeed, metaprogrammers are forced to deal on a continuous basis with both the generative programming code and the generated code. A unified view on both the metaprogramming code and the source code being generated could potentially reduce the cognitive load for the metaprogrammers. Moreover, advancements in programming techniques could be applied simultaneously to both the generative and generated code.

B. Systems Integration and Scalable Metaprogramming

Based on a generic engineering concept, systems integration in information technology refers to the process of linking together different computing systems and software applications, to act as a coordinated whole. Systems integration is becoming a pervasive concern, as more and more systems are designed to connect to other systems, both within and between organizations. Due to the many, often disparate, metaprogramming environments and tools in practice, we argue that systems integration should be explored and pursued more at the metaprogramming level. Just as traditional systems integration often focuses on increasing value to the customer [15], systems integration at the metaprogramming level could provide value to their customers, i.e., the software developers.

Something all implementations of automatic programming or metaprogramming have in common, is that they perform a transformation from domain models and/or intermediate models to code generators and programming code. In general,

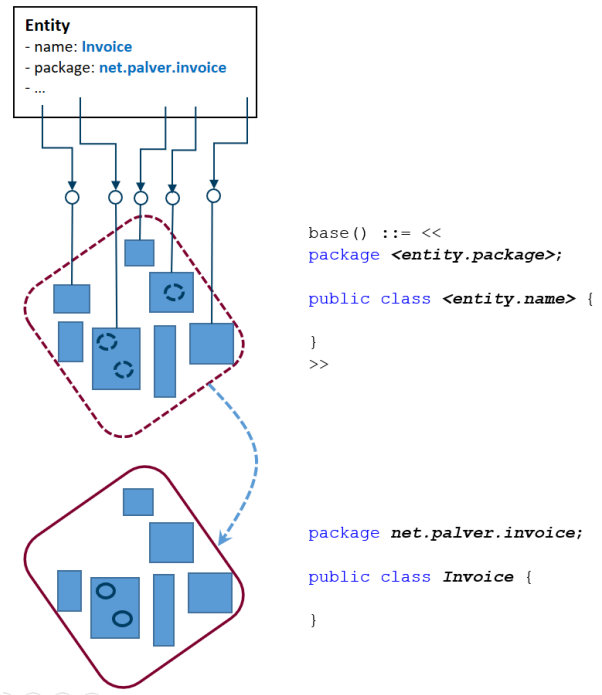


Fig. 2. Instantiating a coding template with model parameters.

metaprogramming or code generation environments also exhibit a rather straightforward internal structure. This structure is schematically represented for a single metaprogramming environment at the left side of Figure 1, and consists of:

- *model files* containing the model parameters.
- *reader classes* to read the model files.
- *model classes* to represent the model parameters.
- *control classes* selecting and invoking the different generator classes.
- *generator classes* instantiating the source templates, and feeding the model parameters to the source templates.
- *source templates* containing the parameterised code.

Figure 2 provides a schematic representation of a very elementary code generation. An instance of a model *entity*, with name *Invoice* and belonging to a package *net.palver.invoice*, is fed into a coding template for a base class. In this template, the values of the model entities are represented as parameters. The generator code will resolve these parameters and replace them with the actual values of the model entity, resulting in real source code for that domain entity.

Another metaprogramming environment will have a similar internal structure, as schematically represented at the right side of Figure 1. Such similar but duplicated architectures exhibit a *vertical integration* architecture. In this architecture, the functional entities are also referred to as *silos*, and metaprogramming silos entail several significant drawbacks. First, it is hard to collaborate between the different metaprogramming silos, as both the nature of the models and the code generators will be different. Second, contributing to the metaprogramming environment will require programmers to learn the internal structure of the model and control classes in the metaprogramming code. As metaprogamming code is intrinsically abstract, this is in general not a trivial task. And third, as contributions of individual programmers will be spread out across the models, readers, control classes, and actual coding templates, it will be a challenge to maintain a consistent decoupling between these different concerns.

We have argued in our previous work that in order to achieve productive and scalable adoption of automatic programming techniques, some fundamental issues need to be addressed [16][4]. First, to cope with the increasing complexity due to changes, we have proposed to combine automatic programming with the evolvability approach of *Normalized Systems Theory (NST)* providing (re)generation of the recurring structure and re-injection of the custom code [16]. Second, to avoid the growing burden of maintaining the often complex metacode and continuously adapting it to new technologies, we have proposed a meta-circular architecture to regenerate the metaprogramming code itself as well [4]. We will go into some more detail on NST and the corresponding metaprogramming environment in the next section.

As this meta-circular architecture establishes a clear decoupling between the models and the code generation templates [4], it allows for the definition of programming interfaces at both ends of the transformation. This should remove the need for contributors to get acquainted with the internal structure of the metaprogramming environment. It also enables a more *horizontal integration* architecture, by allowing developers to collaborate on both sides of the interface. Modelers and designers are able to collaborate on models, gradually improving existing model versions and variants, and adding on a regular basis new functional modules. (Meta)programmers can collaborate on coding templates, gradually improving and integrating new insights and coding techniques, adding and improving implementations of cross-cutting concerns, and providing support for modified and/or new technologies and frameworks. Moreover, an horizontal integration architecture could facilitate collaboration between different metaprogramming environments. Though many trade publications and academic papers on metaprogramming exist, they focus in general on specific implementations and not on the integration of different implementations. Exploring such a collaborative integration is the purpose of the case study in this paper.

C. An Explorative Case Study as a Proof of Concept

Our goal is to investigate the use of an horizontal integration architecture for the collaboration between different metaprogramming environments through an explorative case study. To serve as a representative case study and a valid *proof of concept*, two metaprograming environments were chosen that exhibit several key characteristics. First, these environments themselves are no mere prototypes. They have been developed for years and have been used in practice by many users in many different use cases. Second, these environments target the automatic programming of two totally different types of

software systems: multi-tier web-based information systems, and executable (army) models for simulation systems. Consequently, the two metaprogramming environments have a completely different meta-model. Third, these environments use different technologies at both sides of the horizontal integration architecture, i.e., both the front-end technologies capturing the models, and the target programming languages —even the code templating engines— are different. At the same time however, both metaprogramming environments share a structured decoupling between the definition of models and the generation of code, providing a starting point for an horizontal integration effort.

III. Normalized Systems Elements Metaprogramming

In this section, we present the structure of the metaprogramming environment for web information systems. Its meta-circular architecture explicitly aims to facilitate and realize horizontal integration and scalable collaboration.

Normalized Systems Theory (NST), theoretically founded on the concept of *stability* from systems theory, was proposed to provide an ex-ante proven approach to build evolvable software [16][17][18]. The theory prescribes a set of theorems (*Separation of Concerns*, *Action Version Transparency*, *Data Version Transparency*, and *Separation of States*) and formally proves that any violation of any of the preceding *theorems* will result in combinatorial effects thereby hampering evolvability. As the application of the theorems in practice has shown to result in very fine-grained modular structures, it is in general difficult to achieve by manual programming. Therefore, the theory also proposes a set of design patterns to generate the main building blocks of (web-based) information systems [16], called the *NS elements*: *data element*, *action element*, *workflow element*, *connector element*, and *trigger element*.

An information system is defined as a set of instances of these elements, and the NST metaprogramming environment instantiates for every element instance the corresponding design pattern. This generated or so-called *expanded* boiler plate code is in general complemented with custom code or *craftings* to add non-standard functionality, such as user screens and business logic. This custom code can be automatically *harvested* from within the anchors, and *re-injected* when the recurring element structures are regenerated.

While the NST metaprogramming environment was originally implemented in a traditional metaprogramming silo as represented in Figure 1, it has been evolved recently into a meta-circular architecture [4][5]. This meta-circular architecture, described in [5] and schematically represented in Figure 3, enables both the regeneration of the metaprogramming code itself, and allows for a structural decoupling between the two sides of the transformation, i.e., the domain models and the code generating templates.

In the following subsections, we briefly summarize the different parts of this metaprogramming environment.

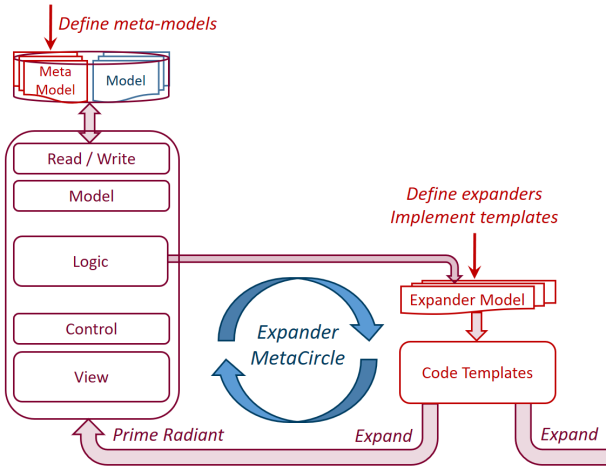


Fig. 3. Closing the meta-circle for expanders and meta-application.

A. Systems Modeling

The domain models for the web-based information systems are specified as sets of instances of the various types of NS elements. As the NS elements, e.g., data and task elements, are closely aligned to traditional primitives in information systems modeling and analysis, their definition and design is similar as well. To this purpose an *NS Modeler* was developed [19], allowing system analysts and designers to enter NS models graphically, in much the same way as traditional modeling and design tools do. Data elements can be modeled in a graphical interface similar to most *ERD (Entity Relationship Diagram)* visualizations and data modeling tools, allowing the designer to define and manipulate data entities, their attributes, and relationships. Task and flow elements can be identified in a graphical interface similar to most *BPM (Business Process Modeling)* visualizations, with the exception that designers are only allowed to define flows as state machines operating on a single target data element [19].

The various NS elements or models can also be designed and manipulated in a dedicated meta-application, called the *Prime Radiant*, using a table-based interface. This meta-application is a regular NS web application that can be generated and rejuvenated based on its own model, being the NS meta-model. Unlike the *NS Modeler*, the *Prime Radiant* allows the designers and developers to specifiy various application and technology settings [5], and to directly invoke both code generation and rejuvenation, and building and deployment of NS web applications.

The domain models of the various NS web applications are stored in XML files specifying the various NS elements, e.g., a data element with its attributes, relationships, and finder queries. The underlying structure of those XML files, i.e., the NST meta-model, is formally defined in corresponding *XSD (XML Schema Definition Language)* schema files. The XML models can be stored both locally and in central repositories, and can be exchanged between between different designers and developers, and between various instances of the NS Modeler

and Prime Radiant. The actual code generation can be invoked from the Prime Radiant meta-application, or from a command line interface accessing the XML files.

B. The NST Meta-Model

As the NS meta-model is just another NS model [4][5], the various elements of the meta-model can be specified in XML files, just like any other instance of a data element. Aimed at the automatic programming of multi-tier *web-based information systems*, the meta-model of the NST metaprogramming environment is a model for web-based information systems. The core part of the data model of this metaprogramming environment is represented in Figure 4 using a screenshot from the *NS Modeler* tool. It is, as mentioned above, similar to most *ERD (Entity Relationship Diagram)* visualizations, but uses colors to distinguish between different types of data entities [19], e.g., light blue for primary data entities and light red for taxonomy entities.

By looking at the NS meta-model, we can browse through the structure of a regular NS model. The unit of an NS domain model is a *component*, and within such a component model, we distinguish the various types of NS elements [16], such as *Data elements*, *Task elements*, and *Flow elements*. These elements, colored light blue and located in the top row, can have options, e.g., *Task options*. Both the entities representing elements and their corresponding options, are accompanied by a typing or taxonomy entity, e.g., *Task element type* or *Task option type*, represented in light red. The data elements contain a number of attributes or *Fields*, where a field can be either a data attribute or a relationship link, and provide a number of *Finders*. Both fields and finders can have options characterized by corresponding option types.

Apart from being more elaborate than the representation of Figure 4, the NS metaprogramming environment also defines a meta-model for the specification of technology settings, such as specific frameworks implementing cross-cutting concerns to be used in the generated code, and build or deployment parameters. In this way, web applications can be generated and deployed using different underlying technologies, while at the same time allowing developers to exchange models with corresponding technology settings to ensure repeatable code generation and deployment of application models.

C. Code Generation

As explained in detail in [5], the NST metaprogramming environment is highly modular and uses a declarative control mechanism. The code generation environment for web-based information systems consists of 182 individual code generators or *expanders*. Every individual code generator or *artifact expander* is declared in an `Expander` XML file. Such an expansion *control file* specifyies for instance the type of element it belongs to, the application layer it belongs to, the technology that it uses, and the various properties of the source artifact that it generates. In this way, the metaprogramming environment can be extended with alternative variations of expanders that provide another implementation and/or use another underlying technology or programming language.

For every declared artifact expander, one needs to provide a coding `Template`, based on the *StringTemplate (ST)* templating engine library. For the NS metaprogramming environment for web applications, the various templates contain currently source code in Java, JavaScript, HTML, XML, and SQL. A template for an individual expander is in general modularized or hierarchically structured itself. To avoid duplication and in accordance with the strategy of of *single sourcing* [20], a template for a *DTO (Data Transfer Object* would use for instance subtemplates for the variable declarations and the get- and set-methods. Other basic coding units like logging or error throwing are also defined in a single template. The templates also contain so-called *anchors*, enabling developers to write additional custom code that can be *harvested* and *re-injected* during consecutive (re)generations.

To access the various attributes and parameters from the elements in the domain models, an XML expander `Mapping` file needs to be defined for every individual expander. Such a mapping file specifies the various parameters that are made available to the template in terms of *Object-Graph Navigation Language (OGNL)* expressions. These expressions are evaluated on the object instances representing the elements of the domain model, e.g., `dataElement.name` [5].

IV. Generative Programming of Simulation Models

In this section, we present the structure of the metaprogramming environment for simulation models. This second metaprogramming environment is concerned with a completely different application domain, i.e., models for simulation systems, and is based on a totally different meta-model. However, by clearly separating the modeling in the front-end from the generative programming in the back-end, it is also pursuing a more horizontal integration architecture.

The United States Army has developed and documented hundreds of approved models for representing behaviors and systems, often separate from the simulation environments where they are to be implemented. The manual translation of these models into actual simulation environments by software developers, leads to implementation errors and verification difficulties, and is unable to avoid the workload of incorporating these models into other simulation environments.

In order to address these potential drawbacks, a generative programming approach is being pursued, aiming to capture military-relevant models within an executable systems engineering format, and to facilitate authoritative models to operate within multiple platforms. The goal of this work is to be able to capture authoritative conceptual models and then to generate software to implement those representations/behaviors. This generated software can be quickly integrated into multiple simulations regardless of their programming language thereby saving development cost and improving the consistency across simulation systems.

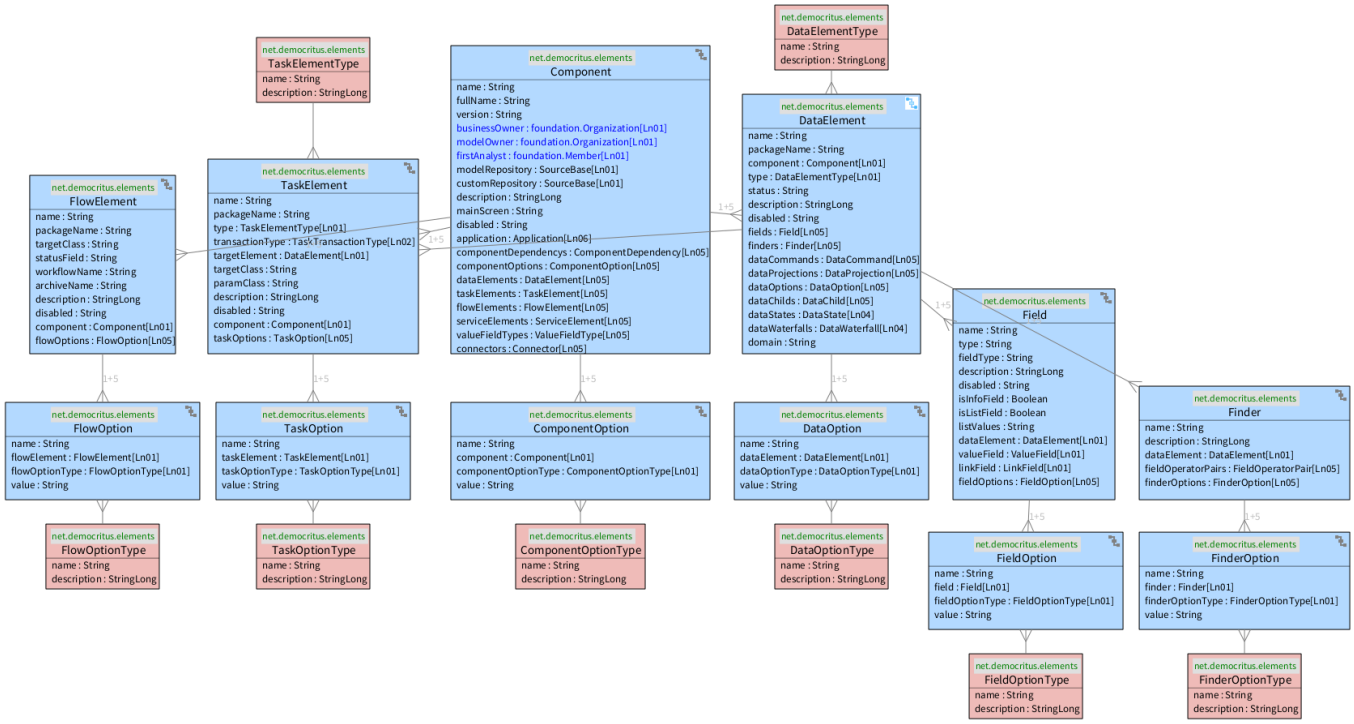


Fig. 4. A graphical representation of the core part the NS (data) meta-model.

The architecture of this metaprogramming environment, schematically represented in Figure 5, divides the problem into two domains, i.e., the front-end and the back-end. In the front-end, corresponding to the conceptual models at the left column, the *Subject Matter Experts (SME)*, scientists, and software model developers are able record the model definitions and behaviors or algorithms. In the back-end, represented in the three other columns, those model definitions and algorithms are transformed through templating and metaprogramming into executable code, targeted at specific architectures and implementations. To properly decouple these parts, an *Interchange Format (IF)* was created that allows one or more front-ends to be created to record models in a way that suits the needs of the front-end user community, and to pass those models to be used for code generation in the back-end.

In the following subsections, we describe the different parts of this generative programming architecture in more detail.

A. Front-End Visual Programming

The Generative Programming environment allows experts to create models using a flow-based programming tool. Flow-based programming [21] has become popular in game engines as well because it allows level designers, artists, and other non-programmers to create some complex business logic. The metaprogramming tool is based on the open-source project *PyFlow* [22], but with many improvements that allow modelers to represent structures and workflows most common within modeling and simulation. The goal of the project is to have a visual tool that allows subject matter experts to author their models without having to know how to develop software within a certain simulation system. The project was initially based on *Blockly* [23], a tool aiming to help non-programmers create software visually. As the structures used in *Blockly* resemble software logic puzzle pieces that fit together in specific ways, the users are fundamentally creating logic in a similar form to software, but without having to know syntax. This mechanism quickly became cumbersome with the more complex models and meant that the authors had to understand some basic software constructs. It was therefore decided to move towards flow-based programming because it was easier to create models as the visual representation of the business logic was based on the flow of data rather than using software constructs. Figure 6 represents a sample node that depicts a function that takes in two variables and outputs the maximum value. An `inExec` pin shows the execution coming into the node along with two other inputs seen on the left, `value1` and `value2`. The line going outwards from the `outExec` pin shows where execution is to go next, while the max line will go to whichever future function that will use that value.

A major concern was to make sure that the selected tool was *representation complete*, i.e., allowed one to represent any business logic that could be constructed in software, provided a rigid structure for inputs and outputs, and was easy to read and write. The selection of PyFlow for the flow-based programming tool allowed us to start with an open-source tool that could be improve upon for the simulation domain. Many additions were made to PyFlow, most notably we have made it more performant for large-scale graphs, added data querying capabilities that are most often used in our simulations, and added the ability for the user to write documentation within

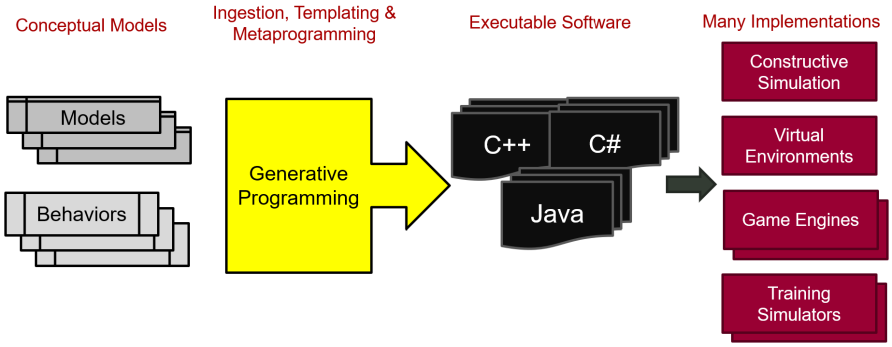


Fig. 5. Schematic representation of the generative programming architecture for simulation models.

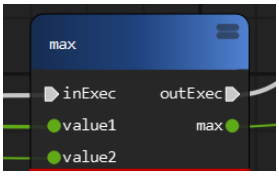


Fig. 6. A flow-based programming node.

the tool and associate that documentation to the nodes and data structures. We have also added an automated capability of exporting a Word document with the documentation and imagery of the graphs, keeping the models self-documenting. By keeping the explanation of how the model is supposed to work within the model development tool (like comments within software), there is more of a chance that the documentation keeps up to date rather than having documents separate from software implementations of the model. In addition to the Word document that gets generated, we also put the documentation text within comments in the generated software making it easier for integration software developers to understand what is being implemented. A screenshot impression of the integrated development environment is represented in Figure 7, showing a standard development environment around the central graph representation.

An important capability of the tool is to allow models to reference data. Datastores are a way to lookup data from a file, database, or potentially a data service. The goal is to make testing with the datastore easy for the user developing the graph, but allow the code generation and developers flexibility in how that data is queried. We created an editor for *Datastore* definitions in the *Types* editor, which allows the user to specify what data is available in the data store, what those types are, and to name and describe them in a typical user interface. The user can also specify *Queries* for that data, specify what the keys are for the lookup, and what values should be returned. Queries result in nodes that can be used in the graph to get results from the datastore as seen in Figure 7.

Once the model developers have created their model and described their data, they can use the visual programming tool to generate software, execute the software, and see the results return from a set of defined inputs. This allows the model developers to ensure that the model is working as they expect before involving simulation developers. Results can be graphed in many ways to visualize outputs. The test results are displayed in the bottom portion of Figure 7. There is a graph of the results that shows two strange oddities in the data. The model developer could recognize these oddities early in the development process and take corrective action before software developers even get involved. Issues can be found in the logic or data while developing the model which results in more accurate models, easier development, and better maintenance of models as logic changes or new data is used.

Finally, the logic and graphs developed in *PyFlow* can be exported to an intermediate format, that can be used to transfer the model from the front-end to the back-end.

B. STE Canonical Universal Format

The interchange format between the front-end and the back-end is based on XML documents, whose structure is defined by an XML Schema or *XSD (XML Schema Definition Language)*. This format structure is called the *Synthetic Training Environment (STE) Canonical Universal Format (SCUF)*.

This meta-model is not intended to support a full programming language, but rather to focus on the domain elements used within the U.S. Army's canonical descriptions of the simulation models. Nevertheless, it represents most concepts of a traditional procedural programming language. Specifically, these include the data type declarations, datastores, and various elements of algorithms, such as conditions, expressions and iterators. Moreover, the XML nature of the format means that it is easily extensible over time as long as the code generation tool is modified accordingly to handle any extended portions.

The logic and graphs developed in *PyFlow* are represented in the SCUF interchange format, and will ultimately be used by the code generation capability to create software. The aim was to provide an intermediary format between the visual development tool and the back-end code generation, in order to separate the two capabilities and to allow other future tools to output SCUF and still take advantage of the code generation capability without having to be compatible with *PyFlow*.

The SCUF meta-model is broken into two parts for ease of depiction. Figure 8 represents the elements related to

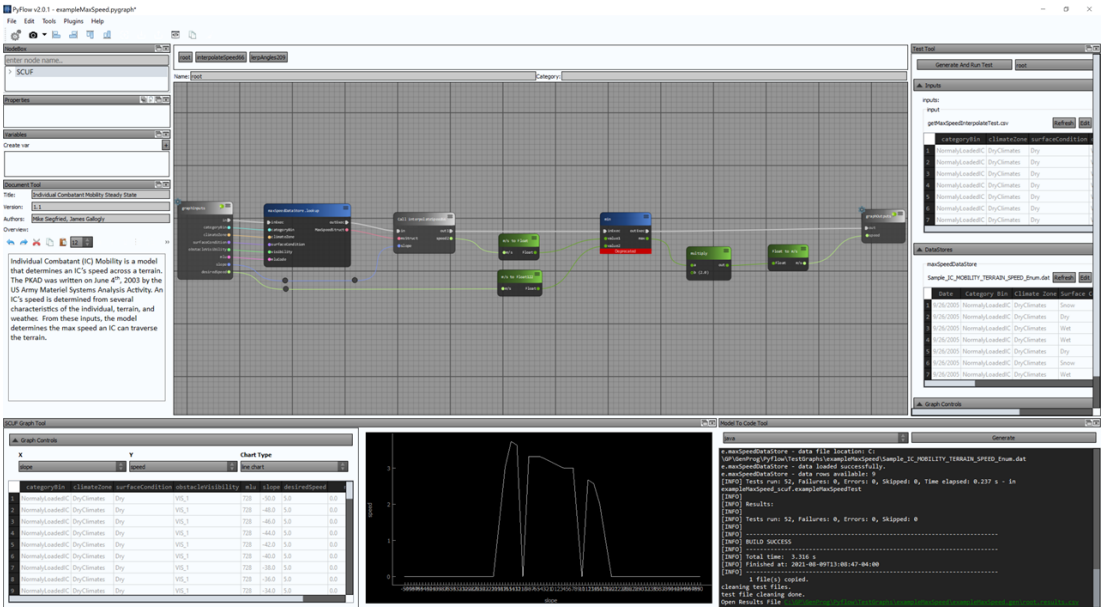


Fig. 7. A screenshot representing the PyFlow-based integrated development environment.

`TypeDefinition` and are static elements such as classes and types. The elements represented in Figure 9 are dynamic elements that describe modeling logic such as inputs, software structures, variables, and outputs. The structure and representation of the SCUF data models is similar to most standard *ERD (Entity Relationship Diagram)* visualizations. An example SCUF output is represented in Figure 10, where you can see the types, including type definitions, enumerations, classes, and datastores at the top of the file and then the dynamic logic in the bottom half of the XML document.

C. Back-End Code Generation

The code generation tool, the *Model To Code Tool (MTCT)* reads the SCUF model files into Java class representations in accordance with the meta-model above. Using the *Visitor pattern* [24] to process the model classes, it traverses the ingested model performing functions without impacting the model classes themselves. Once it is ensured that all dependencies are present and that the model is verified to be correct, the MTCT then uses a templating system to generate software. Currently, the *Apache Velocity* templating engine [25] is used.

Templates have been developed for three different programming languages: C++, C#, and Java. Each meta-model element corresponds to a template which handles the generation for that element. For instance, *ClassType* model elements use the class-template.vm to generate code. To simplify the implementation of these templates, the code generated from any contained elements is referenced in the template using a modular structure. For instance, zero or more *Declare* elements can be contained in a *ClassType*. The code for *Declare* is generated using *declare-template.vm* and passed into the *class-template.vm*. In addition to simplifying the containing element template, this allows changes to low level elements to be implemented in a single location and take effect throughout the generated code. This approach is in accordance with the strategy of of *single sourcing* [20], similar to the NST approach, and reduces the number of templates that need to be implemented to support new languages or simulation environments. As the output from one template feeds into the input of another, we refer to this as a set of *cascading templates*. Figure 11 illustrates a simplified example of this process.

Reuse of existing templates allows us to simplify the process of adding support for new languages. For example, the processing of creating expressions with basic operators or making function calls is the same across multiple languages. MTCT utilizes a search path for locating the individual template files. It will look through the directories in the search path and use the first template it finds with the matching name. Many low-level templates, such as *expression-template.vm* can be reused across languages while specializing those that contain differences, such as *class-template.vm*. The search path for templates also allows users to override certain templates by pre-pending this path with the location of the customized version. In this way, a user could customize how enumerations or even enumerators are generated while reusing all the other existing templates.

Control files determine how the generated code is placed into files and a directory structure. The control files, along with the template search path and cascading templates allow the user to completely control the code generation to create *Architecture Specific Templates (AST)* without modifying the MTCT core functionality. ASTs reduce the amount of integration work and code the developer must write to adapt generated models into a particular simulation system. ASTs are implemented by using Control Files for users to control how source code files are generated from the templates. They

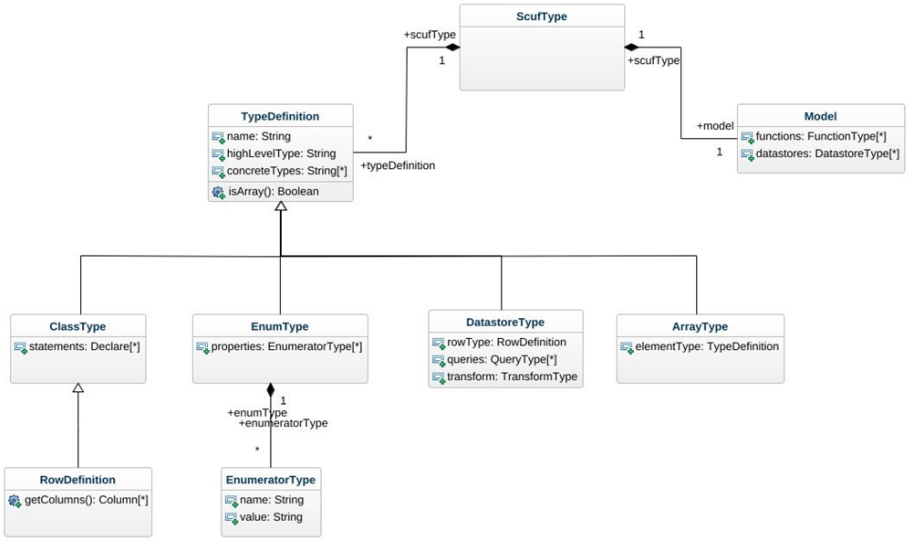


Fig. 8. A graphical representation of the part of the SCUF meta-model related to type definitions.

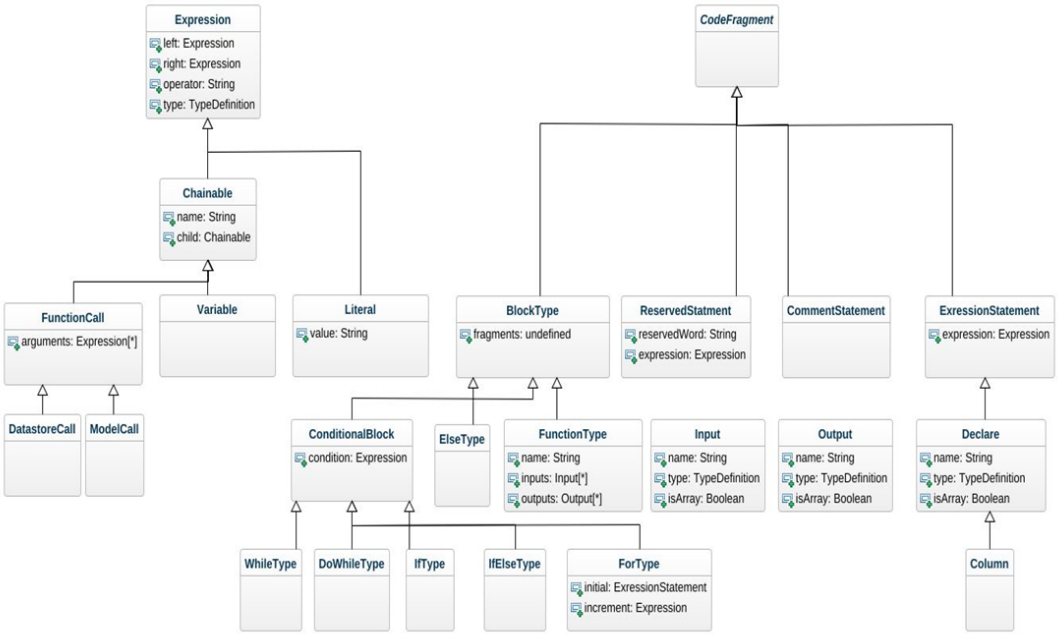


Fig. 9. A graphical representation of the part of the SCUF meta-model related to modeling logic.

allow the user to specify what parts of the SCUF model should be generated and how those parts will be generated. The *Control File* itself is distinct from the templates and specified in JSON format. The file contains a list of sections, each of which can generate code for specific parts of the SCUF. These parts are the enumerations, classes, datastores, and the model. Each section can also use its own template search path. This allows for multiple passes over the meta-model, each potentially with its own configuration. For example, our default implementation for the datastores uses two main templates: one for an interface for the datastore and one for the implementation. The control files specify that the datastore should be passed over twice by the code generator, once with a template for the interface class and once with a template for the implementation class. The cascading template sets simplify this by only requiring the main datastore template to be overridden in this case. Control files also control whether the output for of all model classes of a certain type should be written to a single file or multiple files. A benefit of this approach is allowing for the differences in programming

```
<scuf name="MaxSpeedPKAD" version="Tue Nov 12 2019 09:27:30 GMT-0
  <types>
    <typedef name="degree" highlevelType="number" allowedConcrete
    <typedef name="m/s" highlevelType="number" allowedConcreteTyp
    <typedef name="MLUCode" highlevelType="number" allowedConcret
    <enum name="CategoryEnumBin" highlevelType="enum" allowedConc
      <property name="NormalyLoadedIC" val="13"/>
      <property name="FullyLoadedIC" val="14"/>
    </enum>
    <class name="MaxSpeedStruct">
      <statement name="categoryBin" type="CategoryEnumBin"/>
      <statement name="mluCode" type="MLUCode"/>
      <statement name="-40 Slope Speed" type="m/s"/>
      <statement name="Level Speed" type="m/s"/>
      <statement name="+40 Slope Speed" type="m/s"/>
    </class>
    <datastore highlevelType="datastore" name="maxSpeedDataStore"
      <keyset>
        <key name="categoryBin"/>
        <key name="mluCode"/>
      </keyset>
    </datastore>
  </types>
  <model>
    <function name="getMaxSpeed">
      <inputs>
        <input name="categoryBin" type="CategoryEnumBin"/>
        <input name="mlu" type="MLUCode"/>
        <input name="slope" type="degree"/>
        <input name="desiredSpeed" type="m/s"/>
      </inputs>
      <outputs>
        <output type="m/s"/>
      </outputs>
      <reserved-statement name="return">
        <function-call name="min">
          <arguments>
            <function-call name="interpolateSpeed">
              <arguments>
                <function-call name="lookup_maxSpeedDataStore">
                  <arguments>
                    <variable name="categoryBin"/>
                    <variable name="mlu"/>
                  </arguments>
                </function-call>
                <variable name="slope"/>
              </arguments>
            </function-call>
            <variable name="desiredSpeed"/>
          </arguments>
        </function-call>
      </reserved-statement>
    </function>
  </model>
</scuf>
```

Fig. 10. A sample of a SCUF XML file.

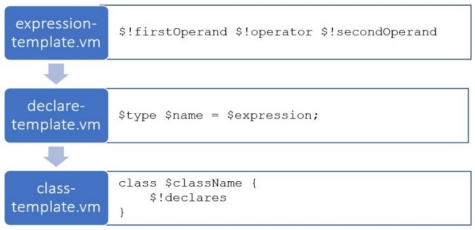


Fig. 11. An example of cascading templating.

languages for how data types and classes may or may not be collocated in the same file. For example, in Java, the classes are expected to be in their own file with the name of the file matching that of the class. As opposed to C++ where the classes can all be included in the same file if the developer chooses to design it that way.

One of the other capabilities of control files are to include external files or *project files* into the code generation process. These files can be whatever the user needs to include in their project such as utility classes or build files. The contents of the file should just be added to a velocity template and then the template and file name just need to be added to the project control section of the control files. These templates also get a default set of information provided by velocity that includes things like the model's name, the list of enumerations, the list of datastores and more. This provides the user with complete flexibility to do anything that they need to do with the project files. The control files simplify the creation of custom template sets for a particular language and simulation environment, allowing adaptation of models to new platforms. Within a control file, the user can specify the initial template search path to be appended to the cascading template search. This allows the user to choose what template set they want to use for each pass over the meta-model. Using control files, external function libraries, and custom template sets, users can write variation of these files to generate code specific to their own architecture, in the same way MTCT customizes generated code for the languages C++, Java, and C#.

Two additional use cases for code generation have been explored: generating a *Unity MonoBehavior* [26] as well as generating behaviors for *RIDE*, the Unity-based simulation simulation environment. To do this, a custom template needs to be written for the main model class, as well as Unity or RIDE specific configuration files in the project control section of the control file. A major challenge related to both architectures is that the models follow a component structure. This means that the models have an internal state that the model itself interacts with and updates. The current generated code supports a more static architecture where all of the inputs are provided as parameters, and it produces the output from a static context. A solution is being investigated where the MTCT would automatically detect which class members in the model need to be included in the component state. This can be done by looking at which variables are being passed into the functions with the *Init* and *Update* tags. These tagged methods represent the methods to initialize a component and to update a component. Doing this will automatically create a component with internal state. These class members will be part of the component and can be handled or used within the glue code as needed. The state of the model would be controlled and customized through the custom template sets.

Another challenge posed by the component modeling in Unity and RIDE are the methods that are meant to handle specific events within the framework. For example, Unity has a start method that runs during the initialization of the component and an update method that runs during each frame. The MTCT needed a way to be aware of these types of methods so that they are customizable and so that they can be handled differently compared to other methods in the model. We are currently looking into a solution where we introduce method tagging within the SCUF. This would allow users to tag certain methods as Update, Start, etc. Doing this would let the MTCT know what type of method it is, and the code generator would be able to handle the different cases. A method having a tag would also let the MTCT know that it is a component style model which would potentially change how the code generation is executed.

V. Toward Integrating the Metaprogramming Environments

We have argued in Section II-C that both selected metaprogramming environments are well suited to be used as part of a representative case study for the horizontal integration of such

metaprogramming environments, and that the structured decoupling between the definition of models and the generation of code, a common characteristic of both models, is a good starting point for this integration. Moreover, the interchange format of the models in both environments is based on XML documents, whose structure is defined by an XML schema.

As the creation of a more horizontal integration architecture to facilitate the collaboration between metaprogramming environments [5] was one of the original goals of the NST metaprogramming environment, it seems logical to initiate this integration effort based on the NST environment architecture. This means that we attempt to map the generative programming environment for simulation models onto the collaboration architecture represented in Figure 3. In this section, we discuss some progress and remaining challenges.

A. Embracing the SCUF Meta-Model

The NST meta-circular metaprogramming environment [5] allows for the structural generation of all reader, writer, and model classes of any model —or meta-model— that can be expressed as a set of NST data elements. The SCUF meta-model, based on XML and defined by an XML Schema, satisfies this requirement. Based on the definition of the SCUF data entities (as represented in the class diagrams of Figures 8 and 9, e.g., *TypeDefinition*, *DatastoreType*, *ConditionalBlock*, *Expression*, *Declare*, *Statement*, etcetera), NST data elements can be created. For instance, *Input* needs to be defined as an NST data element with a *name* field which is a string, a *type* field that is a link to the *TypeDefinition* data element, and an *isArray* field that is a boolean. These data elements can be specified in XML, or in the user interface of the NST meta-application, or even directly generated from the XML Schema. For every data element, the various classes of the NST stack in the left part of Figure 3 can be generated. These include:

- Reader and writer classes to enable reading and writing the XML-based SCUF model files, e.g., *InputXmlReader* and *InputXmlWriter*.
- Model classes to represent and transfer the various SCUF entities, and to make them available as an object graph, e.g., *InputDetails* and *InputComposite*.
- View and control classes to perform *CRUDS (create, retrieve, update, delete, search)* operations in a generated table-based user interface.

This implies that the various existing SCUF models, representing instances of the SCUF data entities and therefore instances of the NST data elements, can be read and made available as an object graph, allowing to evaluate model parameters using *Object-Graph Navigation Language (OGNL)* expressions at the templating engine. Moreover, a NS web application with a table-based user interface can be generated to create, view, manipulate, and write SCUF models.

B. Integrating Modeling and Expansion

The meta-circular architecture of [5] enables the definition of alternative meta-models such as SCUF, and the development of new expanders based on the values and parameters of instances of these new models. Such alternative meta-models can be specified as any regular NS model, both in the NS Modeler and in the Prime Radiant. Upon defining an alternative meta-model such as SCUF, we are currently investigating two modes to provide integrated support for entering actual models based on the new meta-model and expanding these models.

- Based on the new meta-model, a slightly modified NS application is generated, dubbed *Secondary Radiant*, that allows to import/export the actual models from/to XML, and to pass them to the *Prime Radiant*. Importing expanders based on this new meta-model into the Prime Radiant then enables developers to invoke these expanders from the Prime Radiant.
- A runtime kernel, dubbed *Runtime Radiant*, is provided that allows regular NS applications to invoke the templating engine and to evaluate OGNL expressions for arbitrary trees of data objects. In this way, a regular NS application generated based on the new meta-model is able to perform expansion from its actual model data.

Both types of tooling are currently being tested in β-version, and will possibly merge into one solution.

Though conceptually agnostic with respect to different meta-models, a bias toward the web information systems was discovered in the NST metaprogramming environment. Both the invocation of expansion and deployment, and the various technology settings were implicitly linked to the entities *Application* and *Component*. As these enities are specific to web-based information systems, they have been generalized to *ProgramType* and *ModuleType* in the NST environment. At the same time, dedicated meta-elements to specicy various technologies, such as *PresentationLogicSettings*, have been generalized to a generic list of *TechnologyStackSettings*.

C. Streamlining the Control Files

Having defined the SCUF data entities as NST data elements, the NST metaprogramming environment allows to evaluate SCUF model parameters through OGNL expressions in SCUF model graphs, and to make them available to coding templates. In order to simply activate the existing coding templates of the simulation models, and to use the NST metaprogramming environment as a piece of evolvable middleware to pass the SCUF models to the code templates for the simulation models, two tasks remain to be performed at the level of the declarative control.

- Every coding template needs to be declared in a separate XML *Expander* definition.
- For every coding template, the appropriate OGNL expressions to evaluate the relevant model parameters, need to be defined in an XML *Mapping* file.

As the generative programming environment for simulation models has control files as well, they are a solid starting point to create these declarative control files. It is probably even possibe to write software that automates the conversion of

these JSON control files into the XML control files of the NST metaprogramming environment.

D. Supporting the Templating Engine

The fact that both metaprogramming environments use different templating engines causes a final integration issue. A first option would be to convert the *Velocity* templates of the simulation software to the *StringTemplate* format supported by the NST environment. In this scenario, the required effort would be proportional to the template base of the simulation models, and would need to be repeated for integration efforts with other environments using this templating engine. Moreover, *Velocity* templates allow more logic that would have to be ported to Java helper classes in the *StringTemplate* environment.

A second and preferable option is to include support in the NST metaprogramming environment for the *Velocity* templating engine. Considering the limited amount of templating engines being used by metaprogrammers, this scenario seems both manageable and worthwhile. Moreover, the effort would not be proportional to the size of the template base. As there is virtually no logic in the current NST templates, i.e., all model parameters are combined and processed in the software that feeds the templating engine, it is reasonable to say that we expect no major blocking issues.

Important to note is that both metaprogramming environments use modular or so-called cascading templates in accordance with the strategy of single-sourcing. This means that the modular structures of the templates could be preserved identically across the different templating engines.

VI. Conclusion

The automated generation of source code, often referred to as metaprogramming, has been pursued for decades in computer programming, and is considered to entail significant benefits for various disciplines, including software development, systems engineering, modeling, simulation, and business process design. However, we have argued that metaprogramming is still facing several issues, including the fact that it is challenging to realize a scalable collaboration within and between different metaprogramming environments, largely due to the often vertical integration architecture.

In our previous work, we have presented a meta-circular implementation of a metaprogramming environment, and have argued that this architecture enables a scalable collaboration, both within this environment and possibly with other metaprogramming environments. In this paper, we have explored such a collaborative integration with another metaprogramming environment. This second environment for metaprogramming targets the generation of a different type of software systems, and is based on a different meta-model, but exhibits a more horizontal integration architecture as well. This second metaprogramming architecture has been described in detail.

We have shown in this contribution how both metaprogramming environments can be integrated within the proposed meta-circular architecture. We have explained how the generation of the meta-code, i.e., the code that makes the actual parameter models available to the coding templates, can be extended to the second metaprogramming environement, resulting even in tooling that provides integrated support both modeling and expansion or code generation. We have also explained that the only reason that the actual code generation of this second metaprogramming environment cannot be seamlessly integrated yet, is the different format of the generation control files and the use of another templating engine. However, we have also indicated that it should be relatively straightforward to support, possibly even in an automated way, such an alternative control format and/or templating engine.

This paper is believed to make some contributions. First, we show in a constructive way that it is possible to perform an horizontal integration of two metaprogramming environments, and to enable collaboration and re-use between these environments. Such integrations could significantly improve the collaboration and productivity at the metaprogramming level. Moreover, we show that this integration is possible between metaprogramming environments that are based on completely different meta-models, are significant in size, and are being developed and used by application developers on a continuous basis. Second, we explain how the horizontal integration of a second metaprogramming environment with the meta-circular architecture, could largely remove the burden of maintaining the internal classes of such a metaprogramming environment.

Next to these contributions, it is clear that this paper is also subject to a number of limitations. It consists of a single case of integrating a second metaprogramming environment with the meta-circular architecture, although the case deals with two realistic and comprehensive development environments. Moreover, the presented results are still preliminary, and the second metaprogramming environment is not yet operational in the meta-circular architecture, as its control mechanism and templating engine is not yet fully supported in this architecture. Therefore, neither the complete horizontal integration, nor the productive collaboration between the two environments has been completely proven. However, this explorative but nevertheless representative case study can be regarded as an architectural pathfinder, and we have identified some remaining issues that hamper the scalability of the approach.

To further enhance the scalability of the approach, it is imperative to streamline and support the automated exchange of domain models and the corresponding meta-models. We are therefore working on enhanced tooling to allow metaprogrammers to easily define their existing meta-models. Based on these definitions of meta-models, the tooling should be able to extend itself, and to include support for the actual models that are based on these meta-models. The goal is to enable entering, manipulating and viewing these models, and to provide the automatic creation of standardized model data trees and/or control files that can be fed into the various tem-

plating engines. Besides addressing the issues that currently hamper the scalability of the approach, we have also initiated a collaboration with a third metaprogramming environment.

REFERENCES

[1] 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) 2020, 2020, pp. 166–172.

[2] J. R. Rymer and C. Richardson, "Low-code platforms deliver customer-facing apps fast, but will they scale up?" Forrester Research, Tech. Rep., 08 2015.

[3] B. Reselman, "Why the promise of low-code software platforms is deceiving," TechTarget, Tech. Rep., 05 2019.

[4] H. Mannaert, K. De Cock, and P. Uhnak, "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) 2019, 2019, pp. 171–176.

[5] H. Mannaert, K. De Cock, P. Uhnak, and J. Verelst, "On the realization of meta-circular code generation and two-sided collaborative metaprogramming," International Journal on Advances in Software, no. 13, 2020, pp. 149–159.

[6] D. Parnas, "Software aspects of strategic defense systems," Communications of the ACM, vol. 28, no. 12, 1985, pp. 1326–1335.

[7] P. Cointe, "Towards generative programming," Unconventional Programming Paradigms. Lecture Notes in Computer Science, vol. 3566, 2005, pp. 86–100.

[8] K. Czarnecki and U. W. Eisenecker, Generative programming: methods, tools, and applications. Reading, MA, USA: Addison-Wesley, 2000.

[9] L. Tratt, "Domain specific language implementation via compile-time meta-programming," ACM Transactions on Programming Languages and Systems, vol. 30, no. 6, 2008, pp. 1–40.

[10] A. Wortmann, "Towards component-based development of textual domain-specific languages," in Proceedings of the Fourteenth International Conference on Software Engineering Advances (ICSEA) 2019, 2019, pp. 68–73.

[11] K. Gusarovs and O. Nikiforova, "An intermediate model for the code generation from the two-hemisphere model," in Proceedings of the Fourteenth International Conference on Software Engineering Advances (ICSEA) 2019, 2019, pp. 74–82.

[12] C. Mooers and L. Deutsch, "Trac, a text-handling language," in ACM '65 Proceedings of the 1965 20th National Conference, 1965, pp. 229–246.

[13] D. McIlroy, "Macro instruction extensions of compiler languages," Communications of the ACM, vol. 3, no. 4, 1960, pp. 214–220.

[14] J. Reynolds, "Definitional interpreters for higher-order programming languages," Higher-Order and Symbolic Computation, vol. 11, no. 4, 1998, pp. 363–397.

[15] M. Vonderembse, T. Raghunathan, and S. Rao, "A post-industrial paradigm: To integrate and automate manufacturing." International Journal of Production Research, vol. 35, no. 9, 1997, p. 2579–2600.

[16] 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.

[17] 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.

[18] ——, "Towards evolvable software architectures based on systems theoretic stability," Software: Practice and Experience, vol. 42, no. 1, 2012, pp. 89–116.

[19] P. De Bruyn, H. Mannaert, J. Verelst, and P. Huysmans, "Enabling normalized systems in practice : exploring a modeling approach," Business & information systems engineering, vol. 60, no. 1, 2018, pp. 55–67.

[20] K. Ament, Single Sourcing: Building Modular Documentation. Norwich, NY, USA: William Andrew Publishing, 2003.

[21] J. P. Morrison, Flow-Based Programming: A New Approach to Application Development. Van Nostrand Reinhold, 1994.

[22] M. Senthilvel and J. Beetz, "A visual programming approach for validating linked building data," URL: https://publications.rwth-aachen.de/record/795561/files/795561.pdf, 2022, [accessed: 2022-06-15].

[23] B. Rearick, Blockly. Cherry Lake Publishing, 2017.

[24] E. Gamma, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[25] "The Apache Velocity Project," URL: https://velocity.apache.org/, 2022, [accessed: 2022-06-15].

[26] "How to make a video game without any coding experience," URL: https://unity.com/how-to/make-games-without-programming, 2022, [accessed: 2022-06-15].