

Sebastian Böhm
Maxim Vidgof

ZEUS 2025

17th ZEUS Workshop, ZEUS 2025,
Vienna, Austria, 20–21 February 2025
Proceedings

Volume Editors

Sebastian Böhm
University of Bamberg, Distributed Systems Group
An der Weberei 5, DE-96049 Bamberg

Maxim Vidgof
Wirtschaftsuniversität Wien

Copyright ©2025 for the individual papers by the papers' authors.

Copyright ©2025 for the volume as a collection by its editors.

This volume and its papers are published under the Creative Commons License Attribution 4.0 International (CC BY 4.0).

Contents

Alignment of Process Lifecycle and Software Product Line Engineering Phases <i>Philipp Hehne and Manfred Reichert</i>	1
Studying domain dependence in BPMN process modeling: An empirical research proposal <i>Thomas S. Heinze</i>	8
Towards Model Consistency between abstract and explicit Delay-Robustness in Timed Graph Transformation System <i>Mustafa Ghani</i>	12
A Review of Software Architecture Optimization Approaches for Cloud Applications <i>Anton Frisch and Robin Lichtenthäler</i>	16
Seamless Migration of Containerized Stateful Applications in Orchestrated Edge Systems <i>Roman Kudravcev and Sebastian Böhm</i>	25
Comparing Cloud and On-Premises Kubernetes: Insights into Networking and Storage Tooling <i>Jakob Koller and Sebastian Böhm</i>	32
Business Instance Monitoring <i>Lisa Arnold</i>	39
Surgery AI: Multimodal Process Mining and Mixed Reality for Real-time Surgical Conformance Checking and Guidance <i>Aleksandar Gavric, Dominik Bork and Henderik A. Proper</i>	48
Author Index	58

Alignment of Process Lifecycle and Software Product Line Engineering Phases

Philipp Hehnle^{1,*}, Manfred Reichert^{1,*}

¹*Institute of Databases and Information Systems, Ulm University, Germany*

Abstract

Different organisations often run variants of the same business process. As opposed to managing each variant separately, variants can be maintained centrally in a customisable process model, which allows applying changes centrally and propagating them to each variant automatically. However, these approaches focus on the control flow of the business processes. Software Product Line Engineering (SPLE) deals with the efficient development of similar software products by selecting features for individual software variants. In previous work, we applied the concepts of SPLE to business processes in order to be able to select the implementation of certain process variants. For both business process management and SPLE there are a lifecycle and phases, respectively, that describe the development process, which is associated with concepts and tools to facilitate it. When combining process variability on the control flow level with process implementation variability using SPLE, the process lifecycle needs to be aligned with the phases of SPLE. In this work, we present a consolidated lifecycle that allows for the integrated usage of concepts and tools of process variability and SPLE.

Keywords

Business Process Management, Software Product Line Engineering, Process Configuration, Process Variability, Process Family, Software Reuse

1. Introduction

Different organisations run variants of the same business processes. Craftspersons may apply for a special parking permit in various German municipalities (e.g. Munich¹ and Stuttgart²), which permits them to park their cars in the urban area when visiting their clients without acquiring a parking permit for each stop. The process of applying and checking the application for a special parking permit is similar, yet slightly different among the municipalities. Process variants in the public sector have been identified by other researchers as well [1]. Different approaches propose *customisable process models* containing all process variants in one model forming a family of business process variants (process family) [1]. By using a customisable process model, changes and optimisations can be applied centrally, which reduces redundancies and thereby effort [2]. Special modelling languages and extensions to existing modelling languages have been proposed to model variability in business processes and tools have been presented to create a process variant by applying transformations to a customisable process model, which is called *deriving* a process variant [1]. For instance, the *Hide & Block* approach allows hiding and blocking edges of a customisable process model, which removes single edges or entire outgoing paths when deriving a process variant [3]. However, the approaches of customisable process models focus on control-flow variability of the business processes.

To avoid redundancies and thereby reduce costs when developing similar software products the discipline of Software Product Line Engineering has evolved. Software products can be built by selecting features from a set of common core artefacts, which constitute the Software Product Line (SPL) [4]. In literature, selecting features and building a software product is referred to as *deriving* a software product from an SPL [5, 6]. Preprocessors, aspect-oriented programming, special language extensions

ZEUS'25: 17th Central European Workshop on Services and their Composition, February 20 - 21, 2025, Vienna, Austria

*Corresponding author.

✉ philipp.hehnle@uni-ulm.de (P. Hehnle); manfred.reichert@uni-ulm.de (M. Reichert)

🆔 0009-0006-0420-7000 (P. Hehnle); 0000-0003-2536-4153 (M. Reichert)



© 2025 Copyright for this paper by its authors.

¹<https://stadt.muenchen.de/service/info/hauptabteilung-i-sicherheit-und-ordnung-praevention/1072021/>

²<https://www.stuttgart.de/organigramm/leistungen/sonderparkausweise-fuer-gewerbetreibende-und-soziale-dienste.php>

like Jak for Java, and tool chains like FeatureHouse can be used to implement variability and derive software products by selecting and composing features from the common core artefacts of an SPL [6].

A software product that executes a business process in form of a process model is called Process-Aware Information System (PAIS) [7]. In previous work [8, 9], we applied the concepts and tools of SPL Engineering to PAISs to allow for implementation variability of business processes.

1.1. Problem statement

Both the approaches for customisable process models and SPL Engineering have defined steps, which build a lifecycle model and a phases model, respectively. Each step/phase consists of activities and is associated with certain concepts and tools supporting the necessary activities. In order to allow for process variability on the control-flow and implementation level the steps of the lifecycle and the phases of SPL Engineering need to be aligned, i.e. the associated concepts need to be linked and the tools of each step need to be integrated.

1.2. Contribution

This paper presents a consolidated lifecycle for managing and improving process variants in terms of control-flow and implementation variability whose steps are associated with the corresponding concepts and tools.

1.3. Outline

In Section 2, related work is assessed regarding the lifecycle of processes and the phases of SPL Engineering. Section 3 presents a consolidated lifecycle that integrates the steps of the process lifecycle and the phases of SPL Engineering including the concepts and tools of both domains. The paper concludes with a summary and an outlook in Section 4.

2. Related Work

This section presents the phases of SPL Engineering as well as the BPM lifecycle.

2.1. Phases of Software Product Line Engineering

SPL Engineering typically is divided into four phases [6, 10]. Figure 1 shows the phases of SPL Engineering. During *domain analysis*, the requirements in terms of features of the SPL are elicited including the commonalties and differences among the software products resulting in a feature model. Feature models were first introduced by [11] as a tree structure. Figure 5 shows an example feature model. The SPL is the root. The features the SPL consists of are connected via edges to the root. Features themselves can consist of features. Furthermore, features can be marked optional, mandatory, and alternative. During *domain implementation*, the features identified during *domain analysis* are implemented using variability mechanisms (e.g. preprocessor, aspect-oriented programming, language extensions like Jak, tool chains like FeatureHouse) that allow composing them dynamically. *Domain analysis* and *domain implementation* are considered *domain engineering* as they comprise activities concerning the SPL as a whole. When deriving a software product from an SPL, first, during *requirements analysis* the features that shall be contained in the software product are selected from the feature model created during domain analysis. The selected features form a configuration, which is used during *software generation*. Based on the configuration, the implemented features are composed and the software product is thereby generated. *Requirements analysis* and *software generation* are considered *application engineering* as they comprise activities concerning one individual software product that is derived from an SPL.

FeatureIDE is an integrated development environment (IDE) that supports developing SPLs during the corresponding phases [10]. For *domain analysis*, FeatureIDE provides a feature modelling tool (cf.

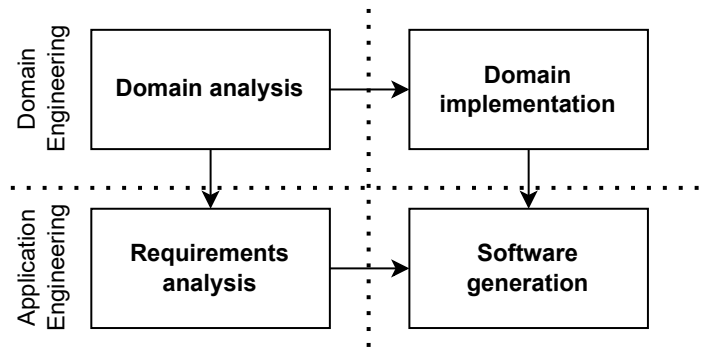


Figure 1: Phases of Software Product Line Engineering adapted from [6]

Figure 5). For *domain implementation*, FeatureIDE support various variability mechanisms. In order to select the desired features during *requirements analysis*, FeatureIDE provides a configuration editor (cf. Figure 6). During *software generation*, FeatureIDE relies on the chosen variability mechanism to compose the software product that shall be derived from the SPL.

2.2. Business Process Management Lifecycle

Business Process Management (BPM) supports “business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information.”[12] Various researchers describe the steps and activities for BPM as BPM lifecycles.

While there are minor differences among the presented BPM lifecycles, the lifecycle models do have a lot in common. Figure 2 shows a consolidated BPM lifecycle that incorporates the commonalties of the presented BPM lifecycle models. During the *process design* step, the processes are modelled [13, 14, 15, 16]. A graphic representation of the business processes facilitates communication for stakeholders. This step is also called *modeling* [17, 18]. In the *process implementation* step, the process is realised in an organisation, usually it is implemented as software [16, 18]. A workflow-management-system can be used. This step is also called *configuration* [13, 14, 15]. In the context of process variability, [17] propose an additional step *selection/instantiation*, in which the desired process variant is selected and automatically instantiated while ensuring consistency and correctness of the instantiated process variant. Although this step is not included by other BPM lifecycle models, we incorporate it in our consolidated BPM lifecycle as we deal with process variants. In the *process enactment* step, the implemented process is executed, often in a workflow engine [13, 14]. This step is also called *execution* [15, 16, 18]. During *process evaluation*, running process instances are monitored and evaluated, data of running processes is aggregated to find deficiencies, and finally improvements are collected, which serve as input for the *process design step* in which the improvements are incorporated into the process model [16, 14]. This step is also called *diagnosis* [13, 15] or *optimisation* [17, 18].

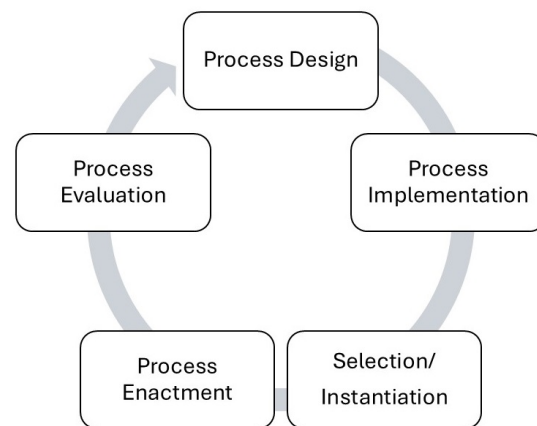


Figure 2: Consolidated BPM Lifecycle

Some researchers like [16, 18] propose further steps, e.g. strategic activities in which organisational and process goals are defined. These steps are out of scope for this work, as we focus on the digitisation

of business processes.

The interested reader is referred to the literature reviews [19] and [20] about BPM lifecycles.

3. A Consolidated Lifecycle for Control-Flow and Implementation Variability of Processes

This section presents a consolidated lifecycle model that maps the phases of SPL Engineering to the steps of the BPM lifecycle in order to allow for process variability on the control-flow and implementation level. First, a real-world example is described based on which requirements are deduced. Finally, the consolidated lifecycle model is presented satisfying the requirements.

3.1. Special Parking Permit for Craftspersons

During a cooperation with German municipalities, the process for checking the special parking permit for craftspersons was inspected. See Example 1 as a simplified description of the process.

Example 1 In various German municipalities, Craftspersons can apply for a special parking permit, which allows them to park in the urban city during their customer visits without acquiring a parking permit for each stop. When a craftsperson has applied for a parking permit, the application needs to be checked. In some municipalities, a clerk checks the application, whereas in other municipalities the application check is carried out automatically. If the application is justified, the parking permit is issued. Otherwise, the craftsperson is notified of the rejection. The notification may be via mail, e-mail, or SMS.

In previous work [8, 9], we implemented Example 1 as a proof-of-concept whereas activities `check application` and `notify craftsperson` are variable, i.e. their implementation may vary. We refer to a digitized business process realised as PAIS with variable activity implementations from which concrete variants may be derived as *PAIS Product Line*. The activity implementations were developed as self-contained plugins. When deriving a variant from the PAIS Product Line, we use the framework and tool chain *FeatureHouse* [21] to compose the selected plugins into one software artefact. Then, the plugins are registered with the corresponding activities they belong to.

Furthermore, it is conceivable that for some municipalities Example 1 is extended in that the craftsperson is notified either way (both when the permit is issued and when the application is rejected). Consequently, the control-flow of the process is variable as in some variants there is a notification activity when a parking permit is issued whereas in others there is not.

The described scenario represents a PAIS Product line whose control-flow is variable as well as its activity implementations.

3.2. Requirements

During the development of SPLs and customisable process models certain steps, which are associated with concepts and tools, need to be carried out, which form the phases of SPL Engineering and the BPM lifecycle, respectively. When implementing a PAIS Product Line comprising a process model with a variable control-flow and activities whose implementations may be exchanged, the steps related to the development of SPLs and customisable process models need to be aligned including the associated concepts and tools, creating a consolidated lifecycle. Consequently, the consolidated lifecycle needs to meet the following requirements:

- R1:* The phases of SPL Engineering and the steps of the BPM lifecycle need to be aligned, i.e. each phase need to be mapped to a step in order to align the related tasks that need do be carried out during the development of an SPL and a customisable process model.
- R2:* The concepts associated with the phases and steps of SPL Engineering and the BPM lifecycle need to be aligned, i.e. need to be connected.

R3: The tools supporting the phases of SPL Engineering and the steps of the BPM lifecycle need to be integrated, so that there is a holistic tool chain facilitating the development of a PAIS Product Line with a variable control-flow and variable activity implementations.

3.3. The Consolidated Lifecycle

In the following, the consolidated lifecycle, which aligns the phases of SPL Engineering and the steps of the BPM Lifecycle, is presented (cf. Figure 3).

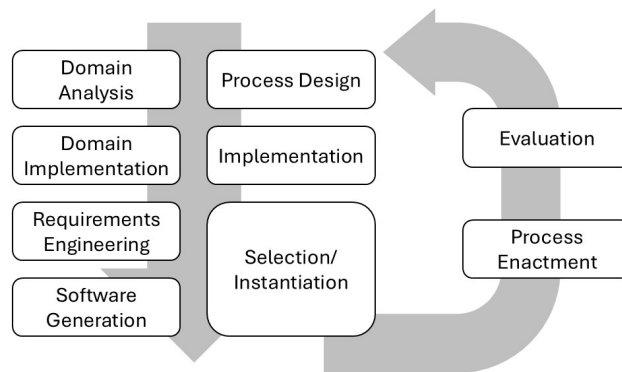


Figure 3: Consolidated Lifecycle for Control-Flow and Implementation Variability of Processes

Process models created in BPMN 2.0 serve as simple means of communication of the represented process for business analysts as well as developers who will implement the process [22]. Besides the visual representation, BPMN 2.0 process models have an XML representation, which allows for the execution of the models. Consequently, the process model, which will be implemented and executed, represents the requirements of the digitized business process. In SPL Engineering, the requirements are elicited during *domain analysis*, which results in a feature model. Consequently, the *domain analysis* can be mapped to the BPM lifecycle step *process design*. In previous work, we also mapped feature models to process models. Each activity, which may be variable (i.e. its implementation may be changed or the entire activity may be optional) equates to a feature in the feature model. Therefore, in previous work [8, 9], we introduced *process feature models*, which represent a PAIS Product Line as a feature model in three levels. The first level corresponds to the PAIS Product Line itself. The second level contains all activities whereas the third level comprises the possible implementations of the activities. Figure 4 shows an example process consisting of a start event, two activities, and an end event. The feature model in Figure 5 represents the example process and is created with FeatureIDE. It can be seen that for both Activities 1 and 2 there are two optional implementations.



Figure 4: Example Process

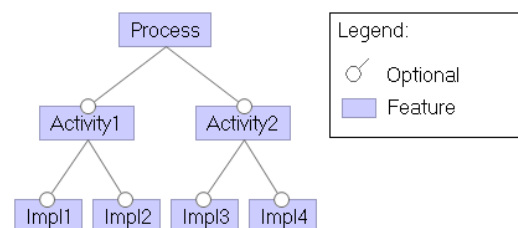


Figure 5: Feature Model in FeatureIDE

In the *domain implementation* phase, the requirements of the SPL are implemented. The BPM lifecycle

has a similar step *implementation* in which the process model, which represents the requirements, is implemented in that it can be executed by a workflow-engine. Thus, *domain implementation* and *implementation* of the BPM lifecycle can be mapped. As introduced in our previous work [9], the activity implementations are developed as composable plugins.

During *requirements analysis*, the desired features are selected from the feature model whereas during the *selection* step of the BPM lifecycle the control-flow of the process is determined, for example by removing optional activities (Hide & Block approach [3]). The configuration editor of FeatureIDE can be used to select the desired implementations for the corresponding activities or if no implementation for an activity is chosen, the activity may be removed. Figure 6 shows the configuration editor in FeatureIDE that allows selecting activity implementations for example process depicted in Figure 4 taking into account the options laid out by the feature model in Figure 5.

The configuration of the selected features from the step *requirements analysis* and *selection* is used during *software generation* and *instantiation* to generate a PAIS including the process model with the selected activity implementation. In line with our previous work [9], the selected implementations may be composed to one artefact using FeatureHouse and register as plugins with the corresponding activities. Furthermore, optional activities for which no implementation was selected are removed from the process model, e.g. by applying the Hide & Block approach.

While the phases of SPL Engineering do not comprise a runtime perspective, the BPM lifecycle considers *process enactment* and *evaluation*. As our consolidated lifecycle model shall cover the development, maintenance, and improvement of PAIS Product Lines, we incorporate the runtime perspective of the BPM lifecycle. During *process enactment*, process instances are started and during *evaluation* the running process instances are monitored for flaws and bottlenecks, which will be analysed and used as input for improvement during the first step *domain analysis* and *process design*.

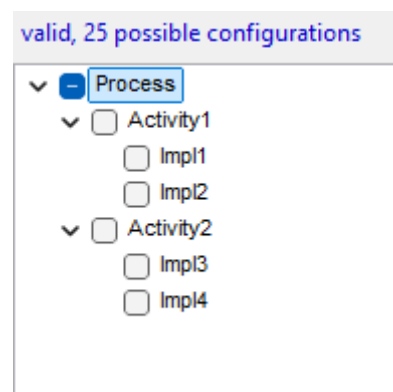


Figure 6: Configuration Editor in FeatureIDE

4. Conclusion

When implementing similar software products as SPL, certain tasks need to be carried out, which involve concepts and tools. These tasks are organised as the phases of SPL Engineering. The BPM lifecycle describes the tasks that need to be carried out in order to develop and maintain process variants on the control-flow level. When implementing process variants whose implementations as well as the control-flow vary, the approaches of SPL Engineering and customisable process models need to be combined. In this work, we presented a consolidated lifecycle model that aligns the tasks of SPL Engineering and the BPM lifecycle. Furthermore, the corresponding concepts were aligned and the tools were integrated for a holistic perspective and usage.

Future work shall reveal in a deep dive how the approaches for customisable process models (e.g. Hide & Block approach) can technically be integrated with the tools of SPL Engineering like the configuration editor of FeatureIDE and be used to automatically derive the process model during *software generation* and *instantiation* when using variability mechanisms like FeatureHouse.

References

- [1] M. La Rosa, W. M. P. van der Aalst, M. Dumas, F. P. Milani, Business Process Variability Modeling: A survey, *ACM Computing Surveys* 50 (2017) 1–45.

- [2] A. Delgado, D. Calegari, F. García, B. Weber, Model-driven management of BPMN-based business process families, *Software & Systems Modeling* 21 (2022) 2517–2553.
- [3] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, M. H. Jansen-Vullers, Configurable Process Models as a Basis for Reference Modeling, in: *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 512–518.
- [4] L. Bass, P. Clements, R. Kazman, *Software architecture in practice*, Always learning, 3. ed., Addison-Wesley, 2013.
- [5] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: A case study, *Journal of Systems and Software* 74 (2005) 173–194.
- [6] C. Kästner, S. Apel, *Feature-Oriented Software Development*, in: *Generative and Transformational Techniques in Software Engineering IV: International Summer School*, Springer, 2013, pp. 346–382.
- [7] M. Dumas, W. van der Aalst, A. Ter Hofstede, *Process-aware information systems: Bridging people and software through process technology*, Wiley, 2005.
- [8] P. Hehnle, M. Reichert, Handling Process Variants in Information Systems with Software Product Line Engineering, in: *2023 IEEE 25th Conference on Business Informatics (CBI), 2023*, pp. 1–10.
- [9] P. Hehnle, M. Reichert, Flexible process variant binding in information systems with software product line engineering, 2024. URL: <https://arxiv.org/abs/2410.17689>. arXiv: 2410. 17689, preprint.
- [10] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, T. Leich, FeatureIDE: An extensible framework for feature-oriented software development, *Science of Computer Programming* 79 (2014) 70–85.
- [11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, S. Peterson, *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report, 1990.
- [12] W. M. P. van der Aalst, A. H. M. ter Hofstede, M. Weske, *Business Process Management: A Survey*, in: *Business process management*, volume 2678 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 1–12.
- [13] W. M. P. van der Aalst, *Business process management: a personal view*, *Business Process Management Journal* 10 (2004).
- [14] M. Weske, *Business Process Management*, Springer, 2019.
- [15] M. Netjes, H. Reijers, W. M. P. van der Aalst, Supporting the BPM life-cycle with FileNet, in: *Proceedings of the 11th International Workshop on Exploring Modeling Methods for Systems Analysis and Design*, 2006, pp. 135–146.
- [16] M. zur Muehlen, D. T.-Y. Ho, Risk Management in the BPM Lifecycle, in: *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 454–466.
- [17] A. Hallerbach, T. Bauer, M. Reichert, Managing Process Variants in the Process Lifecycle, in: *10th Int’l Conf. on Enterprise Information Systems (ICEIS’08)*, 2008, pp. 154–161.
- [18] C. Houy, P. Fettke, P. Loos, Empirical research in business process management – analysis of an emerging field of research, *Business Process Management Journal* 16 (2010) 619–661.
- [19] R. Macedo de Morais, S. Kazan, S. Inês Dallavalle de Pádua, A. Lucirton Costa, An analysis of BPM lifecycles: from a literature review to a framework proposal, *Business Process Management Journal* 20 (2014) 412–432.
- [20] N. Nousias, G. Tsakalidis, K. Vergidis, Not yet another BPM lifecycle: A synthesis of existing approaches using BPMN, *Information and Software Technology* 171 (2024) 107471.
- [21] S. Apel, C. Kastner, C. Lengauer, FEATUREHOUSE: Language-independent, automated software composition, in: *2009 IEEE 31st International Conference on Software Engineering*, 2009, pp. 221–231.
- [22] Object Management Group, *Business Process Model and Notation (BPMN): Version 2.0.2*, 2013. URL: <https://www.omg.org/spec/BPMN/2.0.2/PDF>.

Studying domain dependence in BPMN process modeling: An empirical research proposal

Thomas S. Heinze¹

¹Cooperative University Gera-Eisenach (DHGE), Weg der Freundschaft 4, 07546 Gera, Germany

Abstract

The Business Process Model and Notation (BPMN) language as a de facto standard is omnipresent in business process modeling. This not only applies to its vertical scope, ranging from pen-and-paper process sketches to fully-implemented process automation, but also to its horizontal spectrum, comprising differing domains, e.g., financial services, healthcare, e-government, or webshops. In this paper, we argue for the analysis of differences in the usage of BPMN in varying application domains using an empirical approach: Studying and contrasting features of BPMN models found in process model repositories of varying domains allows us to gain insights into the domains' modeling characteristics. The resulting findings may then be beneficial for defining modeling best practices, supporting future language standardization, or improving (data-driven) modeling tools.

Keywords

mining software repositories, business process modeling, e-government, BPMN

1. Introduction and Motivation

A business process defines a collection of interconnected tasks and activities which allow an organization to achieve a certain goal or objective. Modeling business processes is a crucial step in business process management and its resulting process models form the foundation of the various stages in the business process lifecycle [1] and are thus used for, e.g., documenting, analyzing, or improving business processes. The *Business Process Model and Notation (BPMN)* [2] is a widely accepted business process modeling language, both in industry and academia. BPMN not only supports the various stages of the business process lifecycle and covers the full vertical scope, ranging from sketched process drafts to automated process implementations, but is also the de facto standard in a wide spectrum of application domains, including, e.g., financial or business services, healthcare, e-government, and webshops. As a result, syntax and semantic of the BPMN modeling language is quite crammed and complex, as is illustrated well by the 500 pages of the current BPMN standard language specification [2].

Due to its versatility, we expect different modeling practices and styles in BPMN's day-to-day usage. Empirical research methods like *software repository mining* can help to learn more and understand about how a modeling language is used in practice [3, 4]. Using this knowledge then allows for, e.g., defining modeling guidelines and best practices, supporting future language standardization efforts, or improving modeling tools to better serve their user needs. In prior research, various BPMN language corpora have consequently been established by systematically searching *Github software repositories* and identifying BPMN models [5, 6], yielding extensive data sets comprising thousands of business process models from real open source projects. The resulting data sets were then used to empirically investigate on various research problems, including questions about the standard compliance of the found process models [3, 5], the adaptation of certain design choices and modeling practices [6, 7], or about the frequency of process model duplication and cloning [4, 8]. While this previous work allowed for a better understanding of the general usage of BPMN in open source projects, covering a wide range of differing application domains [4], it remains open whether the obtained findings similarly apply to domains. In particular, the question whether there are not only differences in BPMN's usage across the vertical scope but also across its various application domains, while having been addressed for certain singular domains like healthcare in previous related work [9], still remains open.

ZEUS'25: 17th Central European Workshop on Services and their Composition, February 20–21, 2025, Vienna, Austria

✉ thomas.heinze@dhge.de (T. S. Heinze)



© 2025 Copyright for this paper by its author. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In this position paper, we argue for studying the *domain dependence* of BPMN usage. We believe this to be a worthwhile effort due to various reasons: First, such an analysis helps in providing a more thorough picture of the language’s pragmatics. For instance, certain application domains may impose modeling styles and practices which deviate and even contradict common BPMN usage, though be important when defining modeling guidelines or best practices. Second, analyzing the characteristics of BPMN’s usage in differing domains may help in identifying and prioritizing specific requirements which can be incorporated in future language versions or in defining language subsets or profiles for certain application domains. Third and finally, the advent of large language models (LLMs) promises effective applications of machine learning and data-driven approaches to business process management. This also comprises the use of LLMs for business process modeling, e.g., providing tools for autocompletion of activity labels or other BPMN model structures and transforming process specifications formulated in natural text into matching BPMN process models (cf. [10] for an example in e-government). However, the quality of a LLM’s predictions is influenced by the quality of its training data and the model’s generalization capabilities. This can become problematic if a certain domain features special characteristics, which are otherwise underrepresented in the training data and resulting models. This phenomenon, well-known as out-of-distribution problem [11, 12], may as well apply to the application of LLMs to the process modeling language BPMN across varying application domains.

2. Prospective case study: German E-government

In 2017, the German federal government released a legislation whose primary goal was to provide citizens digital access to 575 selected administrative services and processes over all administrative divisions by the end of 2022, (so-called *Onlinezugangsgesetz (OZG)*). While this objective was not met, the law has been a driver for various e-government initiatives in Germany which also address a more holistic end-to-end digitization of administrative processes. As part of these initiatives, the *Federal Information Management (FIM)* standard¹ is responsible for providing a streamlined methodology for supporting the translation of legal requirements as entailed by law into digital public services implemented by authorities. Due to the federal structure of Germany and since administrative services and processes can thus vary across the different authorities, e.g., federal/state agencies and municipalities, FIM follows a layered approach. At the top layer, master information is derived directly from law and defines administrative services and processes without characteristics of the implementing agency. The reference layer adds technical and organizational aspects, which is for instance required to provide the user-friendly digital services claimed by the OZG. Finally, the local layer allows implementing agencies to add even more details about their respective characteristics, e.g., information about concrete IT services used, etc. In this way, FIM supports reuse of service and process definitions (so-called “one for all” principle), but allows for customization according to the characteristics of implementing agencies.

On a technical note, the FIM standard includes three different modules: (1) *XZuFi*, (2) *XDatenfelder*, and (3) *XProzess*², which comprise XML-based specifications to define administrative services, data fields/forms, and processes, respectively. *XZuFi* defines overall information about administrative services, in particular including a unique identifier (*Leistungskatalog-ID (LeiKa-ID)*), which can be referenced in the other modules. *XDatenfelder* provides uniform structures for data forms and elements together with corresponding plausability rules, which are utilized in the administrative services and build the foundation of web forms for the respective digital services. Eventually, *XProzess* is used to define the processes modeling administrative services. To this end, the *XProzess* notation embeds process models in the BPMN 2.0 language. Note that *XProzess* therefore defines limited subsets of BPMN to be used in process definitions at the master and reference layer: *FIM-BPMN* and *OZG-BPMN*. All three modules are accompanied by libraries and common building blocks. The former provides a repository infrastructure and the latter lowers the implementation burden by providing reusable components for problems which occur frequently and are unspecific to a single administrative service.

¹<https://neu.fimportal.de/>

²<https://www.xrepository.de/details/urn:xoev-de:mv:em:standard:xprozess>

3. Research proposal

We propose the German e-government initiative as a starting point for our investigation into the domain dependence of BPMN usage. As a first step, a corpus of BPMN process models from this application domain has to be established. Mining the respective process repositories, e.g., the FIM master data library³, allows for retrieving XProzess specifications, which can subsequently be parsed to extract BPMN process models. The process models of the resulting data set can then for example be analyzed in conjunction with the more general corpora of open source BPMN models, which have been scraped from Github [3, 5]. In a preliminary analysis, we are then interested in studying and contrasting aspects like process model size, used BPMN features and modeling tools, frequency of model clones, incidence of modeling styles and practices, etc., similar to prior work [6, 8]. A literature survey on the domain dependence usage of BPMN, also within other domains, e.g., healthcare [9], and including domain-specific extensions [13], will complement the insights gained in the preliminary analysis.

As a next step, in a more thorough analysis, certain modeling aspects can be further investigated. For instance, we assume activity labels to be tightly coupled to the application domain. As a result, metrics may emerge for differentiating BPMN process models from different application domains. Inferring such metrics is therefore another research step, which can, e.g., be tackled by training autoencoders that classifies the application domain using the process metrics as input. Note that the mentioned activity labels also play an important role for many process modeling and analysis tools. Algorithms for detecting model clones are for example using activity labels for assessing model similarity [4]. Thus, the tools' performance may be influenced by the application domain. In particular when it comes to tools employing machine learning, as outlined above, the out-of-distribution problem can become an issue. Considering similar experiences with similar tasks for label prediction in conventional programming languages [12], we expect interesting results.

Declaration on Generative AI

The author has not employed any generative AI tools.

Acknowledgments

The author thanks the reviewers for their helpful comments that improved the quality of the paper.

References

- [1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, *Fundamentals of Business Process Management*, Second Edition, Springer, 2018. URL: <https://doi.org/10.1007/978-3-662-56509-4>. doi:10.1007/978-3-662-56509-4.
- [2] Object Management Group (OMG), *Business process model and notation (BPMN), version 2.0.2*, OMG Document formal/2013-12-09, 2014. URL: <https://www.omg.org/spec/BPMN/2.0.2/PDF>.
- [3] T. S. Heinze, V. Stefanko, W. Amme, Mining BPMN processes on github for tool validation and development, in: S. Nurcan, I. Reinhartz-Berger, P. Soffer, J. Zdravkovic (Eds.), *Enterprise, Business-Process and Information Systems Modeling - 21st International Conference, BPMDS 2020, 25th International Conference, EMMSAD 2020, Held at CAiSE 2020, Grenoble, France, June 8-9, 2020*, Proceedings, volume 387 of *Lecture Notes in Business Information Processing*, Springer, 2020, pp. 193–208. URL: https://doi.org/10.1007/978-3-030-49418-6_13. doi:10.1007/978-3-030-49418-6_13.
- [4] M. S. Nikoo, S. Kochanthara, Ö. Babur, M. van den Brand, An empirical study of business process models and model clones on github, *Empir. Softw. Eng.* 30 (2025) 48. URL: <https://doi.org/10.1007/s10664-024-10584-z>. doi:10.1007/s10664-024-10584-z.

³<https://neu.fimportal.de/>

- [5] J. Türker, M. Völske, T. S. Heinze, BPMN in the wild: A reprise, in: J. Manner, D. Lübke, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.), Proceedings of the 14th Central European Workshop on Services and their Composition (ZEUS 2022), Bamberg, Germany, February 24-25, 2022, volume 3113 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 68–75. URL: <https://ceur-ws.org/Vol-3113/paper11.pdf>.
- [6] E. Baalman, D. Lübke, Validation of algorithmic BPMN layout classification (short paper), in: S. Böhm, D. Lübke (Eds.), Proceedings of the 15th ZEUS Workshop, Hannover, Germany, February 16-17, 2023, volume 3386 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 13–20. URL: <https://ceur-ws.org/Vol-3386/paper3.pdf>.
- [7] D. Lübke, D. Wutke, Analysis of prevalent BPMN layout choices on github, in: J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.), Proceedings of the 13th European Workshop on Services and their Composition (ZEUS 2021), Bamberg, Germany, February 25-26, 2021, volume 2839 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 46–54. URL: <https://ceur-ws.org/Vol-2839/paper9.pdf>.
- [8] R. Laue, M. Läuter, Beobachtungen und einsichten zu repositorys von bpmn-modellen, in: J. Michael, M. Weske (Eds.), Modellierung 2024, Potsdam, Germany, March 12-15, 2024, volume P-348 of *LNI, Gesellschaft für Informatik e.V.*, 2024, pp. 157–173. URL: https://doi.org/10.18420/modellierung2024_015. doi:10.18420/MODELLIERUNG2024_015.
- [9] L. Pufahl, F. Zerbato, B. Weber, I. Weber, BPMN in healthcare: Challenges and best practices, *Inf. Syst.* 107 (2022) 102013. URL: <https://doi.org/10.1016/j.is.2022.102013>. doi:10.1016/J.IS.2022.102013.
- [10] S. T. Bachinger, L. Feddoul, M. J. Mauch, B. König-Ries, Extracting legal norm analysis categories from german law texts with large language models, in: H. Liao, D. Duenas-Cid, M. A. Macadar, F. Bernardini (Eds.), Proceedings of the 25th Annual International Conference on Digital Government Research, DGO 2024, Taipei, Taiwan, June 11-14, 2024, ACM, 2024, pp. 481–493. URL: <https://doi.org/10.1145/3657054.3657277>. doi:10.1145/3657054.3657277.
- [11] D. Berend, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, J. Zhao, Cats are not fish: Deep learning testing calls for out-of-distribution awareness, in: 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020, IEEE, 2020, pp. 1041–1052. URL: <https://doi.org/10.1145/3324884.3416609>. doi:10.1145/3324884.3416609.
- [12] B. Gruner, T. Sonnekalb, T. S. Heinze, C. Brust, Cross-domain evaluation of a deep learning-based type inference system, in: 20th IEEE/ACM International Conference on Mining Software Repositories, MSR 2023, Melbourne, Australia, May 15-16, 2023, IEEE, 2023, pp. 158–169. URL: <https://doi.org/10.1109/MSR59073.2023.00034>. doi:10.1109/MSR59073.2023.00034.
- [13] R. Braun, W. Esswein, Classification of domain-specific BPMN extensions, in: U. Frank, P. Loucopoulos, O. Pastor, I. Petrounias (Eds.), The Practice of Enterprise Modeling - 7th IFIP WG 8.1 Working Conference, PoEM 2014, Manchester, UK, November 12-13, 2014. Proceedings, volume 197 of *Lecture Notes in Business Information Processing*, Springer, 2014, pp. 42–57. URL: https://doi.org/10.1007/978-3-662-45501-2_4. doi:10.1007/978-3-662-45501-2_4.

Towards Model Consistency between abstract and explicit Delay-Robustness in Timed Graph Transformation System

Mustafa Ghani

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

Abstract

The increasing interconnectivity of embedded software systems has led to the rise of new types of Multi-Agent Systems, such as Distributed Cyber-Physical Systems, where agents synchronize by exchanging observations and local actions with remote agents. This inter-agent message passing involves transmission, propagation, queuing, and processing delays, which may compromise safety in mission-critical systems due to outdated information. Therefore, we proposed a methodology to derive explicit delay-robust models (resilient to δ -delays) preserving safety from safe abstract models that assume zero-delays. However, this procedure ensures safety at a price of an iterative model checking request. In this paper, we motivate to eliminate the need for costly iterations by exploring behavioral equivalences between explicit and abstract models to define a consistency notion. This consistency facilitates the systematic transfer of verified guarantees to unverified models, effectively eliminating the need for additional model checking.

Keywords

Cyber-Physical Systems Engineering, Formal Modeling, Model Consistency


1. Introduction

The growing interconnectivity of previously isolated embedded software systems has led to the emergence of new types of Multi-Agent Systems, such as Distributed Cyber-Physical Systems (DCPSs). To maintain synchronization in such systems, agents exchange observations and local actions with remote agents. This kind of communication, defined as inter-agent message passing, involve transmission, propagation, queuing, and processing delays [1, 2]. Delays in inter-agent message passing caused by the time elapsed between agents' actions can lead to race conditions or compromise safety requirements in safety-critical systems, as decisions may be based on outdated information. Consequently, software models must clearly differentiate between local, immediate observations (occurring with zero time delay) and remote, δ -delayed observations (requiring up to a specified δ time). In [3], we introduced a methodology to enhance the robustness of zero-delay system models against δ -delays integrated in the rule-based formalism of Timed Graph Transformation Systems. As shown in Figure 1, our approach begins with a given idealized (i.e., assuming zero-delayed inter-agent message passing) safety-critical system model S_{A0} , for which safety has been verified. Based on S_{A0} , we derive a more explicit model

ZEUS2025: 17th Central European Workshop on Services and their Composition, February 20–21, 2025, Vienna, Austria

 mustafa.ghani@hpi.de (M. Ghani)

 0009-0004-6056-2125 (M. Ghani)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

S_{E0} by naive extension of zero to δ -delays for inter-agent message passing. If S_{E0} reveals safety violations, we repair S_{A0} . To accomplish this, we proposed as part of our methodology an automated timed-shift operation to handle δ -delayed messages (called Robustification), resulting in S_{E1} . After verifying safety of S_{E1} , we derive its abstraction, S_{A1} , which must also be verified as safe to ensure it meets the requirements of a safe system model.

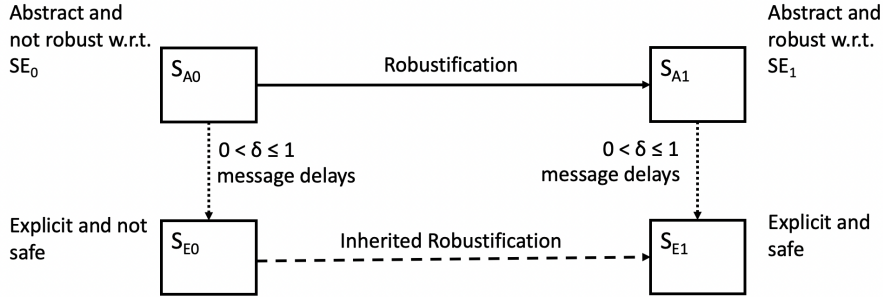


Figure 1: Methodology to derive δ -delayed-robustness for a given zero-delay system model

This approach leverages the modeled information to ensure safety while avoiding unnecessary constraints on the agents' primary behavior. However, since model checking is computationally expensive [4], we aim to make this step for S_{A1} redundant. Therby, we aim to enhance the efficiency of our proposed methodology from Figure 1. To achieve this, we aim to ensure that the verified safety of S_{E1} is inherently carried over to S_{A1} by design.

2. Background

Graph transformation [5] is a formal method in computer science to model dynamic, rule-based changes in (software-intensive) systems. The core idea is to systematically modify a graph structure based on a set of rules. These transformation rules consist of a left-hand side that defines a pattern to match within the existing graph and a right-hand side that specifies how the matched subgraph should be replaced. An additional interface graph can identify which parts of the graph remain unchanged during the transformation [5]. This ensures that transformation is precise and avoids issues like dangling edges. The application of a rule (also called step) may be non-deterministic. A typed graph is a specialized graph in which every node and edge is assigned a type from a predefined type graph. The type graph acts as a schema, specifying the valid types of nodes and edges and their allowed relationships. Extensions like attributed graph transformation allow for richer representations by associating attributes with nodes and edges, enabling computation over more complex structures. To model real-time behavior, Timed Graph Transformation Systems [6] were introduced, where rules are equipped with clocks that first enable and subsequently enforce their application based on clock valuations, which are real values. In this work, we consider typed and attributed Timed Graph Transformation Systems. We refer to [3, 5, 7] for a formal definition of the employed formalism.

3. Research Objective

To bypass model checking of S_{A1} and ease the general proposed methodology, we aim to transfer verified safeness from S_{E1} to S_{A1} by design. The underlying idea is to explore the formal relationship between the two models (i.e., S_{E1} and S_{A1}) to leverage model consistency. Therefore, we define the following research questions.

- Are S_{E1} and S_{A1} formally in relation?
- How can model-based guarantees be systematically transferred from S_{E1} to S_{A1} by design?

To address this research gap, we aim to identify potential behavioral equivalences among S_{A0} , S_{E0} , S_{E1} , and S_{A1} . Establishing such a formal relation may facilitate the transfer of safety guarantees from S_{E1} to S_{A1} by design, thereby eliminating the need for model checking of the latter. However, this approach presents challenges in determining the appropriate level of abstraction required. Furthermore, S_{E1} may introduce potential states that violate safety, which were not reachable in S_{E0} .

4. Related Work

Since model-based consistency research is inherently tied to its domain, and approaches that formally reason about consistency assume additional information about what is being analyzed with respect to the consistency notion [8], we restrict ourselves to models of Timed Graph Transformation Systems with a focus on delay-robustness for mission-critical systems. In [9] the authors presented a different version of Timed Graph Transformation Systems neither supporting quantitative analysis nor considering delay-robustness. In [10, 11], inter-agent message delays were not explicitly considered since message passing was restricted by allowing communication within a given timing interval. In [3], we presented an approach to derive explicit delay-robustness for a given abstract model. However, this approach requires the verification of every generated model (i.e., S_{E0} , S_{E1} , S_{A1}) while in this work we propose a consistency relation making model checking for S_{A1} not required and assuring delay-robustness per design.

5. Conclusion and Future Work

In this paper, we discussed the motivation for reducing the computational cost and the number of model-checking iterations in our previously proposed approach by defining a consistency relation between S_{E1} and S_{A1} . Such a formal relation could serve as the foundation for systematically transferring model-based guarantees. In this context, we identified key research questions and the associated challenges related to achieving this objective.

Acknowledgments

This research was funded by the HPI Research School on Systems Design.

References

- [1] B. A. Forouzan, *Data communications and networking*, Huga Media, 2007.
- [2] M. Van Steen, A. S. Tanenbaum, A brief introduction to distributed systems, *Computing* 98 (2016) 967–1009.
- [3] M. Ghani, S. Schneider, M. Maximova, H. Giese, Deriving delay-robust timed graph transformation system models, in: *International Conference on Graph Transformation*, Springer, 2024, pp. 158–179.
- [4] M. Schmalz, H. Völzer, D. Varacca, Model checking almost all paths can be less expensive than checking all paths, in: *International Conference on Foundations of Software Technology and Theoretical Computer Science*, Springer, 2007, pp. 532–543.
- [5] E. Hartmut, E. Karsten, P. Ulrike, T. Gabriele, *Fundamentals of algebraic graph transformation*, Monographs in theoretical computer science. An EATCS series. Springer (2006).
- [6] S. Neumann, *Modellierung und Verifikation zeitbehafteter Graphtransformationssysteme mittels GROOVE*, Master's thesis, University of Paderborn, 2007.
- [7] M. Maximova, S. Schneider, H. Giese, Interval probabilistic timed graph transformation systems, in: F. Gadducci, T. Kehrer (Eds.), *Graph Transformation - 14th International Conference, ICGT 2021, Held as Part of STAF 2021, Virtual Event, June 24-25, 2021, Proceedings*, volume 12741 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 221–239. URL: https://doi.org/10.1007/978-3-030-78946-6_12. doi:10.1007/978-3-030-78946-6_12.
- [8] R. Pascual, B. Beckert, M. Ulbrich, M. Kirsten, W. Pfeifer, Formal foundations of consistency in model-driven development, in: *International Symposium on Leveraging Applications of Formal Methods*, Springer, 2024, pp. 178–200.
- [9] S. Gyapay, R. Heckel, D. Varró, Graph transformation with time: Causality and logical clocks, in: *Graph Transformation: First International Conference, ICGT 2002 Barcelona, Spain, October 7–12, 2002 Proceedings 1*, Springer, 2002, pp. 120–134.
- [10] M. Maximova, H. Giese, C. Krause, Probabilistic timed graph transformation systems, in: *Graph Transformation: 10th International Conference, ICGT 2017, Held as Part of STAF 2017, Marburg, Germany, July 18-19, 2017, Proceedings 10*, Springer, 2017, pp. 159–175.
- [11] M. Maximova, S. Schneider, H. Giese, Compositional analysis of probabilistic timed graph transformation systems, *Formal Aspects of Computing* 35 (2023) 1–79.

A Review of Software Architecture Optimization Approaches for Cloud Applications

Anton Frisch¹, Robin Lichtenthäler¹

¹*Distributed Systems Group, University of Bamberg, Bamberg, Germany*

Abstract

Developing applications and deploying them on cloud platforms is a common approach. With the continuous evolution of cloud computing and the wide range of technological options, however, making architectural decisions while developing cloud applications can become difficult. One possibility to support the development of cloud applications are software architecture optimization approaches that can evaluate and optimize an architecture according to specific goals. This work provides an up-to-date review of currently existing architecture optimization approaches specifically for cloud applications. Based on the review common optimization goals and approaches are identified and the potential for future work is analyzed.

Keywords

cloud application, software architecture optimization, taxonomy

1. Introduction

Cloud computing is a mature concept that is widely used in the software industry¹. However, since the first cloud offerings have been published, cloud computing has evolved [1] and now a wide range of different platforms and services are available. For developing cloud applications it is thus necessary to make architectural choices that, on top of fulfilling functional requirements, also fit to the inherent characteristics of cloud computing. This means on the one hand taking advantage of cloud computing benefits and on the other hand preparing for inherent issues of cloud environments. Applications specifically built to fit into cloud environments are also called *cloud-native applications (CNA)* [2]. Architecting CNAs means composing a distributed and scalable system out of (micro)services, using cloud-focused design patterns, and operating it on an elastic cloud platform [2]. Providing support for these challenging tasks thus is desirable. One possibility are so-called architecture optimization approaches [3]. These approaches are able to evaluate the architecture of an application according to certain quality goals, propose potential changes and even implement a selected change in an architecture. The overall research field of architecture optimization approaches has been reviewed by Aleti et al. in 2013 [3] together with the development of a taxonomy to classify optimization approaches. However, the main impact of cloud computing has occurred after that and a review of architecture optimization approaches with a specific focus on cloud applications is missing in literature to the best of our knowledge. The aim of this work therefore is to fill this gap. Based on a literature review, the taxonomy of Aleti et al. [3] is adapted to the context of cloud applications and identified approaches are classified based on it. The guiding research questions thus are:

RQ1: How can the current architecture optimization approaches for cloud applications be classified using a structured taxonomy?

RQ2: What is the current state of software architecture optimization research for cloud applications with respect to this classification?

To answer these research questions, based on the foundations presented in Section 2, we outline our methodology in Section 3. The results of applying this methodology are presented in Section 4 and discussed in Section 5, before a final conclusion in Section 6.

17th Central European Workshop on Services and their Composition (ZEUS), February 20 – 21, 2025, Vienna, Austria

✉ anton-liam.frisch@stud.uni-bamberg.de (A. Frisch); robin.lichtenthaeler@uni-bamberg.de (R. Lichtenthäler)

🆔 0000-0002-9608-619X (R. Lichtenthäler)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹<https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/>

2. Cloud Application Software Architectures

Software Architecture is a practice for understanding and managing large-scale structures of software systems [4]. A core foundation for it is the observation that *“it isn’t enough for a computer program to produce the correct outcome. Other software qualities such as dependability and maintainability are also important and can be achieved by careful structuring.”*[4]. Relevant methods include notations to capture system characteristics or technological decisions, techniques to analyze systems, and tools that facilitate and automate these tasks. The specific methods used by software architects, however, depend on the domain and the quality aspects in focus. Aleti et al. [3] had a broad domain scope and thus relied on a more general definition of architecture. According to the ISO42010 standard, an *architecture* is defined as the *“fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes”* [5].

Cloud applications, however, represent a more specific domain for which different aspects may be relevant than for other types of applications. Thus, it is justified to consider cloud application software architectures as a more specific type of architectures for which specific methods can be developed and applied. Important aspects are named by Pahl et al. [6] who define a cloud architecture as *“an abstract model of a distributed cloud system with the appropriate elements to represent not only application components and their interrelationships, but also the resources these components are deployed on and the respective management elements”* [6]. These components, their interrelationships, and the resources on which components are deployed have been described by Kratzke and Peinl [7] in more detail and structured based on a set of layers, from the host layer, over a cluster layer to services and application layers. Implementing so-called cloud-native applications [2] means ensuring certain characteristics for the architecture of an application across the different elements and layers of a cloud system with the goal of ensuring certain quality aspects. Implementing an application in a cloud-native way therefore has the same goal as architecture optimization which is defined by Aleti et al. as *“an automated method that aims to achieve an optimal architecture design with respect to certain quality attributes”* [3].

The key point in architecture optimization is that an automated method is used. This automated method should be able to take a representation of the architecture of an application as an input. On this input, a quality evaluation mechanism has to quantify the current state of an architecture and analyze potentials for optimization. The required changes to optimize an architecture should ideally also be applicable to the application in a structured, automated way. To summarize, the focus of this work is on such automated methods in the specific domain of cloud applications software architectures. That means in this domain a focus is set on the components, typically services, how they communicate and how they are deployed in the cloud.

3. Methodology

The research method for this study adapts the approach from Aleti et al. [3] and the guidelines for systematic reviews by Kitchenham et al. [8]. This section presents the study selection criteria, followed by the search strategy and the data extraction and synthesis process. To select relevant publications, the inclusion and exclusion criteria by Aleti et al. were slightly adapted to focus on the domain of cloud applications as described in Section 2. Three inclusion criteria must be satisfied by each selected publication. Firstly, a machine-processable representation of the software architecture must be provided which the approach receives as input. Secondly, a quality evaluation mechanism must be defined, either to assess relevant quality attributes or ensure that quality attributes and constraints are inherently satisfied. Lastly, degrees of freedom must be defined, meaning that it must be described how the approach can modify a given software architecture to achieve the optimization goal. In addition, works are excluded if they: 1) focus on optimizing a single component in an application without integrating context and interactions with other components; 2) discuss topics not directly related to software architecture, e.g., compiler optimization or hardware-specific optimizations; 3) focus on optimizing hardware instead of software.

The inclusion and exclusion criteria were applied in three stages: a title screening, an abstract review, and a full-text evaluation. In each stage, if for a publication the inclusion and exclusion criteria could not be clearly determined it was moved to the next stage.

The applied search strategy is summarized here and additional information can be found online². As shown in Figure 1, a broad search across Google Scholar, IEEE Xplore, and ACM Digital Library was used to identify initial relevant studies and to guide the further process. This identified seven relevant papers and using the publishing sources, the following four databases were selected for a more targeted search: IEEE Xplore, ACM Digital Library, Elsevier Sciencedirect, and Springer Link. This targeted search applied more specific search strings and revealed additional three relevant studies. Together with two works previously known to the authors, a subsequent backward search using these twelve was done, identifying six more. Finally, a forward search revealed one additional publication, leading to a total of 19 relevant publications that satisfy the inclusion criteria, shown in Table 1. It has to be noted that in relation to the amount of search results, not many publications satisfied all inclusion criteria.

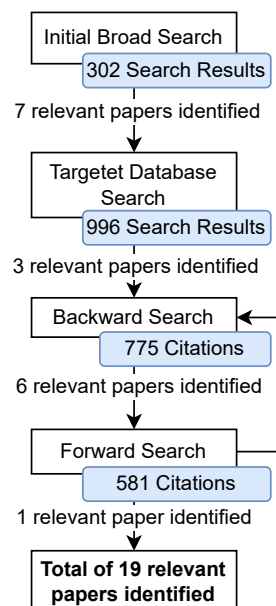


Figure 1: Search Process

Table 1
Identified primary literature for the review

Ref.	Title Authors	Published	Found By	Citing
[9]	Exploring Sustainable Alternatives for Microservices Deployment in the Cloud V. Cortellesa, D. Di Pompeo, M. Tucci	2024, IEEE	Database Search (IEEE Xplore)	
[10]	The μTOSCA Toolchain: Mining, Analyzing, and Refactoring Microservices J. Soldani, G. Muntoni, D., A. Brogi	2021, Wiley	Known Before	
[11]	Modeling and Optimization of Performance and Cost of Serverless Applications C. Lin, H. Khazaei	2020, IEEE	Known Before	
[12]	Architectural Design of Cloud Applications: A Performance-Aware Cost Minimization Approach M. Ciavotta, G. P. Gibilisco, D. Ardagna, E. Di Nitto, M. Lattuada	2020, IEEE	Initial Search (Google Scholar)	[13, 14, 15, 16]
[17]	Performance Optimization of Cloud Applications at Architecture L. X. Du, Y. Ni, P. Ye, X. Wang, R. Xiao	2019, Springer	Database Search (Springer Link)	[18, 15]
[19]	Modeling Optimal and Automatized Cloud Application Deployment S. De Gouw, J. Mauro, G. Zavattaro	2019, Elsevier	Initial Search (Google Scholar)	[20, 21, 22]
[16]	Multi-Objective Optimization of Deployment Topologies F. Willnecker, H. Krcmar	2018, ACM	Initial Search (Google Scholar)	[13, 18, 23]
[24]	ElaClo: A Framework for Optimizing Software Application Topology in the Cloud N. Tankovic, T. G. Grbac, M. zaga	2017, Elsevier	Initial Search (Google Scholar)	[13, 20, 25, 26, 14]
[15]	A Mixed Integer Linear Programming Approach for Multi-Cloud Capacity Allocation M. Ciavotta, D. Ardagna, G. P. Gibilisco	2017, Elsevier	Database Search (ScienceDirect)	[25, 18]
[22]	Zephyrus2: On the Fly Deployment Optimization Using SMT and CP Technologies E. braham, F. Corzilius, E. B. Johnsen	2016, Springer	Backward Search	[25, 21]
[23]	Optimization of Deployment Topologies for Distributed Applications F. Willnecker, H. Krcmar	2016, IEEE	Backward Search	[13, 18]
[14]	Palladio Optimization Suite: QoS Optimization for Component-Based Cloud Apps M. Ciavotta, M. Ardagna, A. Kozirolek	2016, ACM	Backward Search	
[27]	A Model-Driven DevOps Framework for QoS-Aware Cloud Applications M. Guerriero, M. Ciavotta, G. P. Gibilisco, D. Ardagna	2015, IEEE	Initial Search (Google Scholar)	[25, 18]
[26]	A Multi-objective ACS Algorithm for Cost, Performance, and Reliability Optimization A. Ashraf, B. Byholm, I. Porres	2015, IEEE	Forward Search	
[21]	Automated Synthesis and Deployment of Cloud Applications R. Di Cosmo, M. Lienhardt, R. Treinen, S. Zacchiroli, J. Zwolakowski, A. Eiche, A. Agah	2014, ACM	Initial Search (Google Scholar)	
[18]	A Multi-model Optimization Framework for Cloud Applications D. Ardagna, G. P. Gibilisco, M. Ciavotta, A. Lavrentev	2014, Springer	Initial Search (Google Scholar)	[25]
[25]	Search-Based Genetic Optimization for Cloud Software Reconfiguration S. Frey, F. Fittkau, W. Hasselbring	2013, IEEE	Backward Search	
[20]	CloudOpt: Multi-Goal Optimization of Application Deployments J. Z. Li, M. Woodside, J. Chinneck	2011, IEEE	Backward Search	
[13]	PerOpteryx: Automated Application of Tactics in Multi-Objective Software Architecture Optimization A. Kozirolek, H. Kozirolek, R. Reussner	2011, ACM	Backward Search	

²https://github.com/AntonFrisch/Methodology_Architecture_Optimization_Cloud_Applications

To answer **RQ1**, the approaches from the selected publications were firstly categorized according to the existing taxonomy by Aleti et al. [3]. This was done in an extensive descriptive manner without predefined values. In a second step the categorization was reviewed again to refine the classification. Similar concepts were merged together into synonymous terms to create a finite list categorization values. If necessary, new values were created and unused values deleted. Several categories that captured the information in the primary studies on an abstract level were extended for them to capture more specific information. In the final step all papers were reevaluated with the set of finite categories and values created. Therefore employing a test-retest procedure advised by Kitchenham [8] for single researchers. By synthesizing relevant information for every category in a quantitative and descriptive manner from the resulting categorization, **RQ2** is answered.

4. Results

In the following, the results of applying the methodology described in Section 3 are presented with Section 4.1 focusing on **RQ1** and Section 4.2 focusing on **RQ2**.

4.1. A Taxonomy of Software Architecture Optimization Approaches

The overall structure and most categories of the taxonomy by Aleti et al. [3] is kept also for classifying optimization approaches for cloud applications. With the *Problem*, *Solution*, and *Validation* categories, the main aspects of *what* an approach aims to provide, *how* it does so and whether it can be proven to be *valid*, can be described. For the subcategories, however, some modifications were made, based on the reviewed literature. The result is the taxonomy shown in fig. 2. Firstly, the *Domain* category from the original taxonomy, is replaced with the *Architectural Type* category. While the original taxonomy was intended to be applicable within a broad range of domains, cloud applications are a specific domain. And a category which does not provide differentiation between approaches, has no value in a taxonomy. Nevertheless, a differentiation can be done based on which aspects or layers of an architecture are considered. Secondly, subcategories in the *Solution* category have been refined by adding categories that allow for a classification on different layers of abstraction. For example, the *Architecture representation* category enables a classification of the general type of representation. And the *Architecture Representation Method* allows for a classification of which specific language, tool, or technology is used. Thirdly, the largest change is that the specific values assignable in each category are newly defined based on the reviewed approaches. That means values included by Aleti et al. [3] were excluded to enable a clean derivation of relevant values. The resulting values are considered more in detail in the next section.

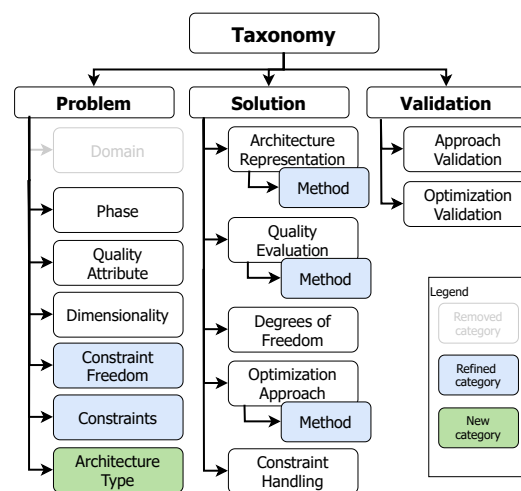


Figure 2: Taxonomy Overview (based on [3])

4.2. Classification of Current Approaches

The result of reviewing the found approaches and their classification based on the taxonomy is shown in table 2. It has to be noted that for the categories *Quality Attribute*, *Constraints*, *Quality Evaluation*, *Degrees of Freedom*, *Optimization Approach*, *Optimization Method*, *Constraint Handling*, and *Optimization Validation* a presented approach could be assigned to multiple values.

Problem Category For the *Phase* in which approaches can be applied, design time approaches are dominant and no approach was identified focusing exclusively on runtime optimization. Thus, these

approaches rely on complete input availability before deployment – either through system modeling or, as in 50% of the approaches, by incorporating runtime data. The most commonly optimized *Quality Attribute* is cost, which is the sole optimization goal in 8 approaches and considered alongside other goals, often performance, in 9 approaches. Other quality attributes appear less frequently. 58% of the reviewed papers use fixed *Constraints*, meaning these approaches adhere to a predetermined unmodifiable set of constraints. In contrast, 37% of the papers allow the user to set the constraints based on the system’s needs. Performance is the most frequently used constraint (74%). In most approaches, multiple constraints are applied, combined in various ways. The *Architecture type* category reveals a clear dominance of deployment architecture approaches, accounting for 95% of the approaches.

Solution Category In the *Architecture representation* category, Performance Models are used the most (84%). A performance model focuses on representing a systems components and their connections combined with performance metrics. In contrast, Architecture models capture the structural organization of components and are used only in two approaches. The specific methods that are used cover a broad range. Extended forms of the PCM (Palladio Component model [29]) are the most prevalent (37%). All other modeling approaches are only used once. The *Quality Evaluation* category classifies how the quality attributes are quantified and evaluated. Model-Based (MB) techniques are predominant (47%), followed by Simulation-Based (SB) approaches (26%). Layered Queuing Networks (LQN) and M/G/1 queuing models emerged as core approaches for predicting how quality attributes such as response time, throughput, and system costs behave under load. LQNs are widely adopted due to their capability to model complex, multi-layered application architectures, with each layer representing different components or services [30]. Approaches are categorized as “Inherent”, if they achieve optimization through inherent solution satisfaction. These approaches inherently satisfy quality attributes and constraints and therefore don’t rely on distinct quality evaluation methods. Regarding *Degrees of Freedom*, Component Allocation stands out as the most frequently applied, (79%). Horizontal Scalability, Component Replication and Resource selection follow closely, indicating a strong emphasis on scaling strategies, that are essential for enhancing flexibility and performance in cloud environments. The combination of Horizontal Scalability, Component Replication, and Component Allocation is notably prevalent, appearing together in 11 approaches [27, 18, 12, 21, 24, 16, 9, 17, 22, 15, 26]. For the *Optimization strategy* approximate methods dominate in the findings (79%). Exact methods are less common (42%). Within these, Mixed-Integer Linear Programming and Tabu Search are the most prevalent, highlighting their effectiveness in solving specific types of architectural optimization problems. Meanwhile, Genetic Algorithms and Evolutionary Algorithms make up a significant portion of the approximate methods. In the reviewed approaches, Tabu Search is frequently used as a second phase refinement method following an initial solution generated by Mixed Integer Linear Programming. Genetic Algorithms can be considered as a subfield of Evolutionary Algorithms but since their frequent appearance during the review they were categorized separately. For *Constraint Handling*, the prohibit method is the most common. It ensures that only feasible configurations are generated by strictly prohibiting any solutions that violate predefined constraints. Repair strategies are the second most common. They allow minor adjustments to solutions to meet constraints.

Validation Category In the *Approach Validation* category case studies are often used to demonstrate practical applicability in realistic settings. However, only a few case studies are conducted within a productive industrial setting [24, 19] and most are using available open-source applications as examples [25, 17] or prototypes [27, 10, 9]. Experiments are also widely used, allowing for controlled, theoretical validation in simulated environments [18, 15, 20]. Finally, Benchmark Problems, specifically the SPEC-jEnterpriseNEXT benchmark, used by Willnecker et al. [23, 16], are less common. For *Optimization Validation*, a significant portion (53%) of studies did not present any explicit comparative validation. Comparison with a baseline is otherwise the predominant method, providing a benchmark against other simpler algorithms or approaches.

Table 2

Resulting classification according to the taxonomy

1.1 Problem Category

Phase	
Design Time (84%)	[18, 12, 21, 24, 28, 16, 10, 15, 9, 17, 22, 20, 23, 14, 13, 26]
Hybrid (16%)	[27, 19, 25]
Quality Attributes	
Cost (89%)	[27, 18, 12, 21, 24, 11, 16, 19, 15, 9, 17, 22, 20, 14, 13, 25, 26]
Performance (47%)	[24, 11, 16, 9, 17, 23, 13, 25, 26]
Resource Utilization (11%)	[16, 23]
Software Quality (5%)	[10]
Power Consumption (11%)	[9, 20]
SLA Violations (5%)	[25]
Reliability (5%)	[26]
Dimensionality	
Single-Objective Optimization (58%)	[27, 18, 12, 21, 11, 19, 10, 15, 22, 23, 14]
Multi-Objective Optimization (37%)	[24, 16, 9, 17, 13, 25, 26]
Hybrid (5%)	[20]
Constraint Freedom	
Fixed (58%)	[18, 24, 11, 9, 17, 20, 23, 14, 13, 25, 26]
Customizable (37%)	[27, 12, 21, 16, 19, 15, 22]
None (5%)	[10]
Constraints	
Performance (74%)	[27, 18, 12, 24, 11, 16, 15, 9, 17, 20, 23, 14, 13, 25, 26]
Resource Limits (47%)	[27, 18, 12, 21, 16, 19, 15, 22, 20]
Deployment Constraints (32%)	[21, 16, 19, 22, 23, 25]
Cost (42%)	[24, 11, 16, 9, 17, 13, 25, 26]
Power Consumption (5%)	[9]
License Availability (5%)	[20]
SLA Violations (5%)	[25]
Reliability (5%)	[26]
None (5%)	[10]
Architecture Type	
Deployment (95%)	[27, 18, 12, 21, 24, 11, 16, 19, 15, 9, 17, 22, 20, 23, 14, 13, 25, 26]
Software Architecture (5%)	[10]

1.3 Validation Category

Approach Validation	
Case study (58%)	[27, 12, 21, 24, 19, 10, 9, 17, 14, 13, 25]
Experiment (32%)	[18, 11, 15, 22, 20, 26]
Benchmark problems (11%)	[16, 23]
Optimization Validation	
Not presented (53%)	[27, 21, 11, 16, 19, 10, 9, 23, 14, 13]
Comparison with baseline heuristic algorithm (42%)	[18, 12, 24, 15, 17, 20, 25, 26]
Comparison with random search (5%)	[24]
Internal comparison (5%)	[22]

1.2 Solution Category

Architecture Representation			
Performance Model (48%)		[27, 18, 12, 21, 11, 16, 19, 15, 9, 17, 22, 23, 14, 13, 25, 26]	
Architecture Model (11%)		[24, 10]	
Evaluation Model (5%)		[20]	
Architecture Representation Method			
PCM [extended] (42%)		[27, 18, 12, 16, 15, 23, 14, 13]	
ACM [extended](11%)	[21, 22]	ATG (5%)	[24]
Directed Graph (5%)	[11]	ABS extended (5%)	[19]
TOSCA (5%)	[10]	UML extended (5%)	[9]
CAPOM (5%)	[17]	LQM (5%)	[20]
KDM (5%)	[25]	None (5%)	[26]
Quality Evaluation			
Model-Based (47%)		[27, 18, 12, 11, 15, 9, 20, 14, 13]	
Simulation-Based (26%)		[24, 16, 19, 23, 25]	
Nonlinear Math. Function (21%)		[27, 18, 12, 15]	
Inherent (26%)		[21, 10, 17, 22, 26]	
Quality Evaluation Method			
Layered Queueing Network (42%)		[27, 18, 12, 15, 9, 20, 14, 13]	
M/G/1 (21%)		[27, 18, 12, 15]	
Palladio-bench (11%)		[16, 23]	
MC-OQN (5%)	[24]	Analytical Model (5%)	[11]
ABS Simulator (5%)	[19]	CDOSim (5%)	[25]
Inherent (26%)		[21, 10, 17, 22, 26]	
Degrees of Freedom			
Component Allocation (79%)		[27, 18, 12, 21, 21, 24, 16, 19, 9, 17, 22, 20, 23, 14, 13, 25, 26]	
Horizontal Scalability (74%)		[27, 18, 12, 21, 24, 16, 19, 15, 9, 17, 22, 23, 14, 13, 25, 26]	
Component Replication (74%)		[27, 18, 12, 21, 21, 24, 16, 19, 15, 9, 17, 22, 20, 23, 13, 26]	
Resource Selection (58%)		[27, 18, 12, 21, 24, 19, 15, 17, 22, 14, 25, 26]	
Vertical Scalability (26%)		[11, 19, 23, 13, 25]	
Component Selection (16%)		[19, 14, 13]	
Provider Service Selection (11%)		[15, 25]	
Software Pattern (5%)		[10]	
Workflow Orchestration (5%)		[11]	
Optimization Strategy			
Approximate (79%)		[27, 18, 12, 24, 11, 16, 10, 15, 9, 17, 23, 14, 13, 25, 26]	
Exact (42%)		[27, 18, 12, 21, 19, 15, 22, 20]	
Optimization Method			
Mixed Linear Int. Prog. (26%)		[27, 18, 12, 15, 20]	
Tabu Search (26%)		[27, 18, 12, 15, 14]	
Genetic Algorithm (26%)		[9, 14, 13, 25]	
Constraint Programming (16%)		[21, 19, 22]	
Evolutionary Algorithm(16%)		[24, 16, 17, 23]	
Greedy Algorithm (5%)		[11]	
Refactoring Rules (5%)		[10]	
Ant Colony Optimization (5%)		[26]	
Constraint Handling			
Prohibit (79%)		[27, 18, 12, 21, 24, 11, 19, 15, 9, 22, 23, 14, 13, 25, 26]	
Repair (37%)		[27, 18, 12, 16, 15, 17, 14]	
None (5%)	[10]	Penalty (5%)	[20]

5. Discussion

Architecture Optimization Approaches for Cloud Applications, as considered here, remain a niche topic, reflected by the relatively few papers found in the literature search. Although this observation may in part be the result of the more strictly formulated inclusion criteria in Section 3 that consider only approaches presenting a structured and automated optimization method. More approaches are available which however include manual steps or do not provide as concrete optimization suggestions as the approaches considered in this work.

To answer **RQ1**, we can state that Aleti et al.'s taxonomy already enables an effective categorization of cloud application optimization approaches. We adapted it by removing one category, adding new categories with different abstraction levels and defining a new set of values as described in 4.1.

For **RQ2** our findings in 4.2 represent the basis from which the following conclusions can be drawn: **Dominance of Design-Time Approaches** Most reviewed approaches focus on optimization before deployment, allowing early performance and cost predictions without incurring additional test infrastructure expenses. This design-time emphasis supports more complex solution derivation, as runtime constraints (e.g., adaptation speed) are less critical. However, questions remain about the actual benefits of runtime optimization versus predefined scaling policies.

Prevalence of Cost and Performance as Key Quality Attributes: Cost and performance dominate in the reviewed approaches. Cost optimization, in particular, appears in nearly all single-objective approaches and all multi-objective ones, reflecting the economic focus of using cloud services. Furthermore, performance and cost are more straightforward to observe at runtime in order to validate optimization approaches. Optimization approaches for other quality aspects, like maintainability or portability require more effort in their validation, since case studies or experiments need to be done in a longer time span. While this focus on performance and cost is practical, it highlights a research gap concerning other quality attributes, for example also reliability.

Focus on Performance Models and Palladio Component Model Many approaches rely on performance models, with PCM being the most widely used thanks to its maturity and tool support. This strong focus indicates PCM's effectiveness in predicting and evaluating QoS attributes. Although alternatives such as Aeolus or customized models exist and are employed the PCM is the only model with widespread adoption in the reviewed approaches.

Dominance of Approximate Optimization Strategies 79% of the approaches apply heuristic or approximate approaches to tackle the NP-hard [31] nature of cloud architecture optimization. Exact methods appear less frequently and often in combination with approximations, usually by limiting problem scope or mixing modeling techniques to ensure feasibility.

Validation Gap in Cloud Architecture Optimization. Few standardized benchmarks exist, making comparisons across approaches difficult. Most validations rely on case studies rather than real-world industrial settings, and lack detailed demonstrations of economic benefits. Future research could focus on robust benchmarking frameworks and well-documented success stories to strengthen the field's credibility and practical impact.

Some limitations of this review must be acknowledged. Only complete optimization approaches presented in academia were included which are typically toolchains comprised of different subfields like software modeling and the formulation of optimization problems. Therefore advancements in individual subfields and industry-driven solutions are not part of this study. Additionally, threats to completeness arise from the selection of database and search terms. Also the review was mainly conducted by a single researcher. Therefore the risk of some bias can't be completely excluded, but was mitigated through a clear methodology and taxonomy-based extraction.

As stated before, we are not aware of another review for software architecture optimization approaches specifically targeting cloud applications. Therefore, as related work mainly the reviewed primary studies would have to be considered and the review by Aleti et al. [3] upon which this study is built.

6. Conclusion

This work presents a systematic review of 19 papers on architecture optimization approaches for cloud applications. A taxonomy for categorizing approaches was derived and current research trends in the field were identified. Key findings include the dominance of design-time optimization, cost-focused goals, reliance on performance models like PCM, and the prevalence of approximate optimization methods. Gaps in validation strategies and a lack of standardized benchmarks were identified, highlighting areas for improvement. These insights aim to guide future research and development for new approaches, advancing the field with more robust and versatile solutions.

References

- [1] A. Taherkordi, F. Zahid, Y. Verginadis, G. Horn, Future cloud systems design: Challenges and research directions, *IEEE Access* 6 (2018) 74120–74150. doi:10.1109/access.2018.2883149.
- [2] N. Kratzke, P.-C. Quint, Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, *Journal of Systems and Software* 126 (2017) 1–16. doi:10.1016/j.jss.2017.01.001.
- [3] A. Aleti, B. Buhnova, L. Grunske, A. Koziolak, I. Meedeniya, Software architecture optimization methods: A systematic literature review, *IEEE TSE* 39 (2013) 658–683. doi:10.1109/tse.2012.64.
- [4] M. Shaw, P. Clements, The golden age of software architecture, *IEEE Software* 23 (2006) 31–39. doi:10.1109/ms.2006.58.
- [5] ISO/IEC/IEEE, ISO/IEC/IEEE 42010:2022 - Software, systems and enterprise – Architecture description, 2022. URL: <https://www.iso.org/standard/74393.html>.
- [6] C. Pahl, P. Jamshidi, O. Zimmermann, Architectural Principles for Cloud Software, *ACM Transactions on Internet Technology* 18 (2018) 1–23. doi:10.1145/3104028.
- [7] N. Kratzke, R. Peinl, ClouNS - a Cloud-Native Application Reference Model for Enterprise Architects, in: *IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, IEEE, 2016, pp. 1–10. doi:10.1109/edocw.2016.7584353.
- [8] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, techreport EBSE-2007-01, Keele University and Durham University, 2007. URL: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf.
- [9] V. Cortellessa, D. Di Pompeo, M. Tucci, Exploring Sustainable Alternatives for the Deployment of Microservices Architectures in the Cloud, in: *2024 IEEE 21st International Conference on Software Architecture (ICSA)*, 2024, pp. 34–45. doi:10.1109/ICSA59870.2024.00012.
- [10] J. Soldani, G. Muntoni, D. Neri, A. Brogi, The μ TOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures, *Software: Practice and Experience* 51 (2021) 1591–1621. doi:10.1002/spe.2974.
- [11] C. Lin, H. Khazaei, Modeling and Optimization of Performance and Cost of Serverless Applications, *IEEE TPDS* 32 (2021) 615–632. doi:10.1109/TPDS.2020.3028841.
- [12] M. Ciavotta, G. P. Gibilisco, D. Ardagna, E. D. Nitto, M. Lattuada, M. A. A. Da Silva, Architectural Design of Cloud Applications: A Performance-Aware Cost Minimization Approach, *IEEE Transactions on Cloud Computing* 10 (2022) 1571–1591. doi:10.1109/TCC.2020.3015703.
- [13] A. Koziolak, H. Koziolak, R. Reussner, PerOpteryx: Automated application of tactics in multi-objective software architecture optimization, in: *Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures*, ACM, Boulder Colorado USA, 2011, pp. 33–42. doi:10.1145/2000259.2000267.
- [14] M. Ciavotta, M. Ardagna, A. Koziolak, Palladio Optimization Suite: QoS optimization for component-based Cloud applications, in: *9th EAI International Conference on Performance Evaluation Methodologies and Tools*, ACM, Berlin, Germany, 2016. doi:10.4108/eai.14-12-2015.2262562.
- [15] M. Ciavotta, D. Ardagna, G. P. Gibilisco, A mixed integer linear programming optimization

- approach for multi-cloud capacity allocation, *Journal of Systems and Software* 123 (2017) 64–78. doi:10.1016/j.jss.2016.10.001.
- [16] F. Willnecker, H. Krmar, Multi-Objective Optimization of Deployment Topologies for Distributed Applications, *ACM Transactions on Internet Technology* 18 (2018) 1–21. doi:10.1145/3106158.
- [17] X. Du, Y. Ni, P. Ye, X. Wang, R. Xiao, Performance Optimization of Cloud Application at Software Architecture Level, in: K. Li, W. Li, H. Wang, Y. Liu (Eds.), *Artificial Intelligence Algorithms and Applications*, volume 1205, Springer Singapore, 2020, pp. 724–738. doi:10.1007/978-981-15-5577-0_58.
- [18] D. Ardagna, G. P. Gibilisco, M. Ciavotta, A. Lavrentev, A Multi-model Optimization Framework for the Model Driven Design of Cloud Applications, in: C. Le Goues, S. Yoo (Eds.), *Search-Based Software Engineering*, volume 8636, Springer International Publishing, Cham, 2014, pp. 61–76. doi:10.1007/978-3-319-09940-8_5.
- [19] S. De Gouw, J. Mauro, G. Zavattaro, On the modeling of optimal and automatized cloud application deployment, *Journal of Logical and Algebraic Methods in Programming* 107 (2019) 108–135. doi:10.1016/j.jlamp.2019.06.001.
- [20] J. Z. Li, M. Woodside, J. Chinneck, M. Litoiu, CloudOpt: Multi-goal optimization of application deployments across a cloud, in: *7th International Conference on Network and Service Management*, 2011, pp. 1–9. URL: <https://ieeexplore.ieee.org/abstract/document/6103947>.
- [21] R. Di Cosmo, M. Lienhardt, R. Treinen, S. Zacchiroli, J. Zwolakowski, A. Eiche, A. Agahi, Automated synthesis and deployment of cloud applications, in: *29th ACM/IEEE International Conference on Automated Software Engineering*, ACM, Vasteras Sweden, 2014, pp. 211–222. doi:10.1145/2642937.2642980.
- [22] E. Abraham, F. Corzilius, E. B. Johnsen, G. Kremer, J. Mauro, Zephyrus2: On the Fly Deployment Optimization Using SMT and CP Technologies, in: M. Franzle, D. Kapur, N. Zhan (Eds.), *Dependable Software Engineering: Theories, Tools, and Applications*, Springer International Publishing, Cham, 2016, pp. 229–245. doi:10.1007/978-3-319-47677-3_15.
- [23] F. Willnecker, H. Krmar, Optimization of Deployment Topologies for Distributed Enterprise Applications, in: *12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*, IEEE, Venice, 2016, pp. 106–115. doi:10.1109/QoSA.2016.11.
- [24] N. Tankovic, T. Galinac Grbac, M. Zagar, ElaClo: A framework for optimizing software application topology in the cloud environment, *Expert Systems with Applications* 90 (2017) 62–86. doi:10.1016/j.eswa.2017.07.001.
- [25] S. Frey, F. Fittkau, W. Hasselbring, Search-based genetic optimization for deployment and reconfiguration of software in the cloud, in: *35th International Conference on Software Engineering (ICSE)*, IEEE, San Francisco, CA, USA, 2013, pp. 512–521. doi:10.1109/ICSE.2013.6606597.
- [26] A. Ashraf, B. Byholm, I. Porres, A Multi-objective ACS Algorithm to Optimize Cost, Performance, and Reliability in the Cloud, in: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, IEEE, Limassol, Cyprus, 2015, pp. 341–347. doi:10.1109/UCC.2015.54.
- [27] M. Guerriero, M. Ciavotta, G. P. Gibilisco, D. Ardagna, A Model-Driven DevOps Framework for QoS-Aware Cloud Applications, in: *17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE, Timisoara, Romania, 2015, pp. 345–351. doi:10.1109/SYNASC.2015.60.
- [28] M. Loukides, *The Cloud in 2021: Adoption Continues*, O’Reilly Media, Inc., 2021. URL: <https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/>.
- [29] S. Becker, H. Koziolok, R. Reussner, The Palladio component model for model-driven performance prediction, *Journal of Systems and Software* 82 (2009) 3–22. doi:10.1016/j.jss.2008.03.066.
- [30] G. Franks, T. Al-Omari, M. Woodside, O. Das, S. Derisavi, Enhanced Modeling and Solution of Layered Queueing Networks, *IEEE TSE* 35 (2009) 148–161. doi:10.1109/TSE.2008.74.
- [31] G. Canfora, M. Di Penta, R. Esposito, M. L. Villani, An approach for qos-aware service composition based on genetic algorithms, in: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO ’05*, Association for Computing Machinery, New York, NY, USA, 2005, p. 1069–1075. doi:10.1145/1068009.1068189.

Seamless Migration of Containerized Stateful Applications in Orchestrated Edge Systems

Roman Kudravcev^{1,*}, Sebastian Böhm¹

¹University of Bamberg, An der Weberei 5, Bamberg, 96047, Germany

Abstract

Edge and fog computing has established an innovative approach in the distributed systems context and enhanced traditional cloud computing by improving latency, bandwidth utilization, and data protection. To orchestrate such environments, dynamic changes in load, like the number of edge devices, must be considered and often result in the need to migrate services to other nodes. This also requires the seamless migration of highly available stateful services to handle such changes. In this paper, we propose a tool for seamless service migration while addressing critical issues like state management, networking, and service availability. We perform a real-world experiment and quantitatively evaluate resource utilization, response time, and availability during the migration and idle states of the involved Kubernetes clusters. We show that it is possible to provide a tool that almost achieves a seamless migration experience with high availability to enable changes for orchestrated edge environments.

Keywords

Edge Computing, Kubernetes, Service Migration, Workload Migration, Orchestration

1. Introduction

Edge and fog computing have emerged as innovative distributed computing paradigms, extending traditional cloud infrastructure capabilities. They aim to meet the growing demand for real-time, latency-sensitive applications and data processing closer to the source, such as Internet of Things (IoT) devices [1]. These paradigms significantly enhance traditional cloud infrastructure by improving latency, bandwidth utilization, and data protection, enabling efficient real-time applications [2]. Cloud-edge orchestration is essential for managing workloads effectively across cloud, edge, and IoT layers. Ensuring efficient application placement, reducing latency, and optimizing resource usage is critical. By dynamically assigning workloads and making policy-driven decisions, orchestrators enable scalability, fault tolerance, and seamless operation, even in complex, distributed environments [3]. Despite its importance, cloud-edge orchestration faces several challenges, including service placement. Service placement is a key challenge in edge and fog computing. It involves determining the optimal deployment of services or applications within a distributed network to minimize latency, energy consumption, and bandwidth usage while maximizing resource availability and ensuring network reliability [4]. However, dynamic changes in load, the number of edge devices, and their locations often require services to be migrated to alternative nodes to maintain performance [5]. This migration is especially challenging for certain applications that cannot tolerate downtime. For example, applications that require continuous availability or state persistence during migration (stateful applications).

Currently, stateful migration and request forwarding are not implemented or evaluated during the migration process, although other studies have considered service placement and migration of stateless applications. Also, there is no comprehensive literature on this particular problem. Hence, this paper's objective and contribution are designing, implementing, and validating a tool for seamless application migration in edge and fog computing environments. To address this gap, the proposed tool integrates self-developed methods and existing technologies to address critical issues such as state management,

17th Central European Workshop on Services and their Composition (ZEUS), February 20 - February 21, 2025 in Vienna, Austria

*Corresponding author.

✉ roman.kudravcev@stud.uni-bamberg.de (R. Kudravcev); sebastian.boehm@uni-bamberg.de (S. Böhm)

🌐 <https://github.com/romankudravcev> (R. Kudravcev); <https://www.uni-bamberg.de/en/pi/team/boehm-sebastian/> (S. Böhm)

🆔 0000-0003-3719-1923 (S. Böhm)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

networking, and service availability. A key aspect of this research is benchmarking and experimental evaluation. This is done to assess the performance and effectiveness of the tool during migration. The benchmarking process measures system availability, resource utilization, and migration time. This comprehensively analyzes the tool's impact before and during migration.

We take a Design Science Research (DSR) approach. We focus on developing and evaluating a migration tool for edge orchestration environments. The tool is implemented to address key challenges such as state management and availability. Its effectiveness is validated through quantitative benchmarking. Metrics such as availability, response time, resource utilization, and migration duration are measured to assess its efficiency and impact on system performance.

The rest of the paper is organized as follows: First, Section 2 discusses existing approaches to migration in cloud- edge orchestration and the current state of research. After that, we cover the concepts of the tools used for the migration tool (Section 3), which helps to understand how the individual components work. Then, we examine the implementation of the migration tool and investigate the individual problems that need to be solved for a successful migration (Section 4). Section 4 will also look at these problems and explain which tools are used to solve them and how they are used. We will then evaluate the migration tool, looking at predefined metrics and comparing them to the system in an idle state (Section 5). Finally, we review the tool and the evaluation (Section 6) and outline plans for further experimental studies (Section 7).

2. Related Work

Ma et al. [6] propose a framework for efficient live migration of edge services using Docker containers, using their layered storage architecture to minimize the amount of data transferred during the migration. However, the work focuses on single-container migrations. It does not address orchestrated multi-container setups or stateful applications that require state management.

Similarly, Kaur et al. [7] investigate live migration of containerized microservices across Kubernetes (K8s) clusters. Their solution ensures uninterrupted communication with the migrated services using Traefik ingress controllers and DNS redirection. While effective, this approach relies on manual configuration and primarily targets stateless workloads. It leaves gaps in automating migration for stateful applications and handling dynamic resource management in orchestrated environments.

This research addresses these limitations with an automated tool for stateful and stateless migration in orchestrated edge systems. The proposed solution focuses on seamless state management and real-time request forwarding to ensure minimal downtime and consistent performance during migrations.

3. Background

3.1. Migration

Migration generally encompasses moving data, applications, or systems from one environment to another. Typical scenarios include moving workloads between cloud platforms, data centers, or storage systems. In edge computing, workload migration is used to offload workloads from cloud data centers to edge nodes to improve latency and bandwidth for end users. Particularly stateful and stateless application migration are in focus. Different strategies for workload migration are available, with benefits and downsides regarding migration time and performance degradation, as shown by [8].

Each strategy requires careful planning to address compatibility, security, and minimizing downtime. This holds explicitly when multi-tier applications, consisting of database (DB)s and caches, must be moved with minimal downtime.

The previously mentioned stateful application stacks store information about past interactions, such as session data or DB records. To ensure continuity and proper functionality, migrating stateful application stacks involves transferring both the application and its state. This includes maintaining service availability while migrating DBs and session data [9].

On the other hand, stateless applications retain no information about previous interactions and treat each request independently. Migrating stateless applications is more manageable because only the application itself needs to be transferred, with no state synchronization or data migration required [10].

3.2. Tunneling

Tunneling is a networking technique that allows secure data to travel over public networks intended for private use. It creates a direct connection between two networks by encapsulating data packets, allowing them to traverse networks that do not natively support the original protocol. This encapsulation also supports encrypted communications to ensure data security.

Tunneling is commonly used in Virtual Private Networks (VPNs) to bypass firewalls, support unsupported protocols, and establish secure connections. It simplifies communication between networks without requiring extensive configuration or routing through multiple servers [11, 12].

However, tunneling has its drawbacks. Encapsulation consumes resources, which can slow down communication. In addition, while packets are encrypted, tunnels bypass firewalls. This poses a security risk if unauthorized access is gained. Proper management is essential to mitigate these vulnerabilities [13, 14].

4. Implementation

4.1. Networking

A secure connection is inevitable for the migration process. Both environments must communicate to replicate the DB to the target environment. Since the DBs may contain sensitive data, unauthorized external access must be prohibited. Therefore, we want to use a tunneling tool to connect container orchestration platform clusters. We solved this problem with Submariner¹. Submariner is an open-source project that enables seamless networking between Pods and Services across multiple K8s clusters, regardless of whether running on-premises or in the cloud. It provides cross-cluster Layer 3 (L3) connectivity using encrypted or unencrypted connections. This is realized with a tunnel using VXLAN. This allows workloads in different clusters to communicate as if they were on the same network. Submariner is designed to be network plug-in (CNI) agnostic to ensure compatibility with various K8s networking setups.

After migration, handling requests sent to the source environment is critical, often because IoT devices have not yet been updated to point to the target environment. We assume that devices that receive a response from the origin environment will also receive information about the target for subsequent requests, as similarly discussed in [15]. To ensure uninterrupted service, we implemented a mechanism to forward requests from the origin to the target environment, assuming the target has the current DB state and primary role. The solution uses an HTTP reverse proxy deployed in the source environment. This proxy intercepts HTTP requests, replicates their details (method, headers, body), and forwards them to the target environment. It then returns the target response to the client, preserving headers and status codes for transparency. Although designed for HTTP, this approach can be adapted for TCP or UDP traffic. The proxy forwards raw byte streams for TCP, and for UDP, it forwards datagrams. This flexible forwarding mechanism ensures seamless communication across protocols.

4.2. Migration

To enable the migration of our system, we developed a tool written in the programming language Go. This tool scans the resources within a K8s cluster, such as deployments, services, ingress routes, config maps, and secrets. It then checks whether these resources already exist in the target environment. If

¹<https://submariner.io/>

they are not, the tool cleans up the resources by removing unique identifiers (e.g., creation dates and IDs) before applying them to the new cluster.

For data migration, we focus on SQL DBs, specifically PostgreSQL, which is the most popular SQL DB.² In container orchestration platforms like K8s, SQL DBs are typically deployed using stateful sets or operator-managed DB clusters. Operator-managed clusters handle tasks such as high availability, scaling, rolling updates, resource management, and security, making them a robust choice for database management.³

We used the CloudNativePG³ operator for testing. Many PostgreSQL operators, including CloudNativePG, support bootstrapping a new DB from an existing one. During migration, the target DB cluster is bootstrapped as a replica of the source DB, which continues to run as the primary DB. Once the target environment is synchronized with the source, applications, and data are fully migrated. At this point, the roles of the clusters must be switched: the replica is promoted to primary, and the original primary is demoted to the replica. This ensures the target environment can handle writes without errors since replicas typically do not allow writes. In CloudNativePG, this process includes synchronized demotion and promotion of DB clusters.³ The process is similar for PostgreSQL, deployed in a StatefulSet, but requires additional manual steps. First, we enable logical replication on the source DB so the target can replicate it. Next, we enable the publishing of changes on the source. The DB schema is then imported from the source, and the target DB subscribes to the source's publication. Once replication is complete, we can switch to the target DB by disabling the subscription.

5. Evaluation and Results

5.1. Experimental Setup and Design

To evaluate the migration's impact on the environments and key metrics, we designed a controlled experimental setup to ensure that the results were reproducible and consistent.⁴ Therefore, we used two Ubuntu 20.04 Virtual Machines (VMs) with 2 vCPUs, 4 GB memory and an SSD with a capacity of 30 GB each. Both VMs run on-premises on one physical host machine with Kernel-based Virtual Machine (KVM) as hypervisor and containerd as container runtime. These VMs were split into two single-node clusters: a origin cluster for migration and a target cluster as the destination.

The setup consists of three components: a resource utilization collector, a test application, and a client application. The resource utilization collector used K8s' Metrics API⁵ to monitor cluster-wide CPU and memory usage each second, storing the data with timestamps in an SQLite DB. The test application is a message store with a REST API for storing, retrieving, and deleting messages. It used a PostgreSQL DB managed by the CNPG operator. A message contains a time stamp, a unique ID, and a message so we can later check which message may not have been received. The client application is a lightweight HTTP client that simulates an IoT device and is configured to send requests to the test application's REST API. The client supports adjustable request rates and GET/POST ratios. It logs the details of each request, namely HTTP method, message content, success status, timestamp, and response time. After all requests have been sent, the client retrieves the stored messages from the test application to determine message loss, availability, and average response time. A GET request is considered as failed if the client receives an HTTP status code outside the 200 range. The number of failed POST requests is determined by looking if the DB contained the unique message sent by the client.

Our evaluation consists of two scenarios. The first scenario measures the idle load without migration to establish a baseline for performance and resource utilization under normal conditions. The second scenario evaluates the load during migration. By testing these two scenarios, we can compare performance and resource utilization between the two scenarios. We can also observe and evaluate

²<https://survey.stackoverflow.co/2024/technology#1-databases>

³<https://cloudnative-pg.io/>

⁴Tool and resources for the performed experiment available online: <https://github.com/romankudravcev/clustershift-benchmark>

⁵<https://kubernetes.io/docs/tasks/debug/debug-cluster/resource-metrics-pipeline/>

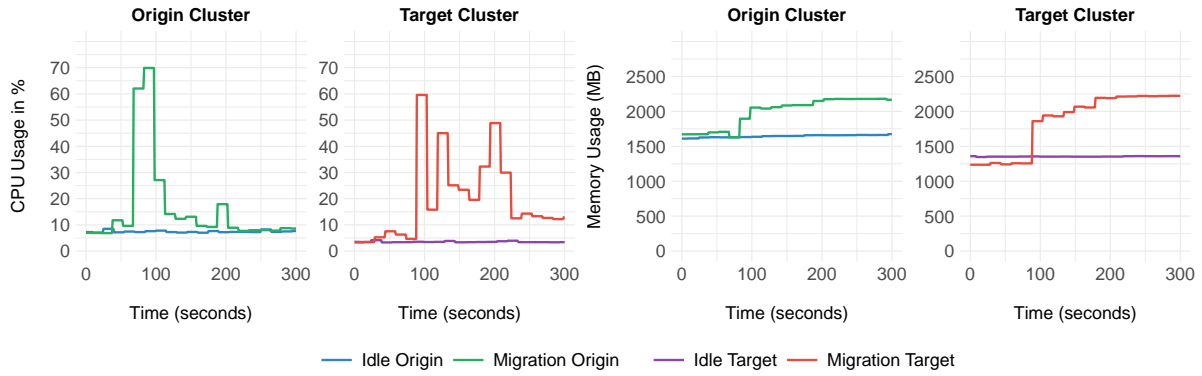


Figure 1: CPU and Memory Utilization of the Origin (left) and Target (right) K8s Clusters.

availability and downtime during a migration. We deploy the resource utilization collector and the test application on that cluster and start making requests with our implemented client. For testing during migration, we deploy the resource utilization collector on both clusters to get an overview of the resource utilization of both clusters and deploy the test application on the origin environment. Then, we start making requests with our implemented client and start our migration process.

5.2. Experimental Results

Figure 1 shows the CPU and memory utilization of the origin and target clusters during the idle and migration states. Idle corresponds to normal cluster load with no migration components, while migration includes deploying the migration components and executing the migration process. In the origin cluster, at about 30 seconds, a small spike in memory usage can be observed. This reflects the deployment of the Submariner Broker and Operator, which requires about 30 MB additional memory. A similar increase can be observed in the target cluster at roughly 40 seconds. The CPU utilization also experiences an increase due to the deployment of the Submariner resources. It increases from approximately 6.85% to a peak of 11.75% on the source cluster and from 3.45% to a peak of 7.55% on the target cluster. At 100 seconds the most significant jump in both CPU and memory utilization is noticed, caused by the replication of the PostgreSQL DB. A memory spike of 430 MB on the origin cluster and 605 MB on the target cluster is observed. Both clusters experience a peak memory load of approximately 2200 MB, which remains until the end of the experiment. On the CPU side, the origin cluster peaks at 70% utilization, while the target cluster peaks at 60%. Both peaks are also caused by the replication of the DB. After the completion of the migration process, the replica DB is promoted to primary, and the original primary is demoted and decoupled from the target DB. This results in a drop in the CPU utilization to about 8% on the origin cluster and 13% on the target cluster.

Figure 2 shows the response time of our deployed test application comparing idle and migration states for GET and POST requests. In the idle state, the average response time for GET requests is 34.5 ms ($\sigma = 2.83$). While migration, the average response time for a GET request increases to 35.9 ms ($\sigma = 6.16$). For POST requests, the average response time during idle is 133.47ms ($\sigma = 69.76$), while during migration, it decreases slightly to 118.17ms ($\sigma = 74.23$). Notably, the migration process causes a considerable number of outliers during migration for GET requests.

To evaluate availability and downtime during the migration we check the number of failed requests and their timestamps. A total of 2165 requests were sent, with approximately 70% being POST requests (1515) and 30% (650) being GET requests. Out of 650 GET requests, 12 failed (1.85%). For POST requests, 29 out of 1515 failed (1.91%). Overall 41 out of the 2165 sent requests failed, resulting in an availability of 98,11%. The total downtime during the migration was 4.66 seconds. This was calculated by looking at the timestamps of failed requests.

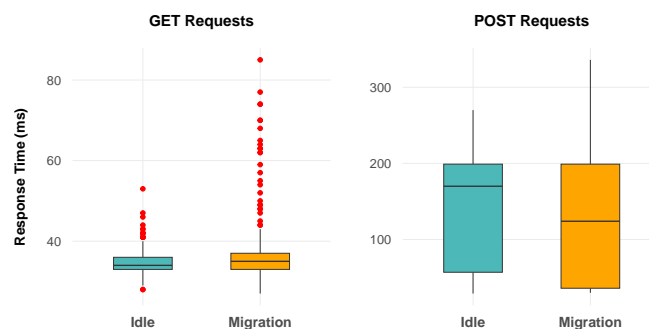


Figure 2: Response Time of the Test Application during Idle and Migration States for GET and POST Requests.

6. Discussion

The results show that Submariner has minimal impact on CPU and memory utilization, supporting its use in migration scenarios. DB replication, on the other hand, caused more significant spikes in resource usage. This is expected because replication requires additional resources to transfer large amounts of data and ensure data consistency. However, it is important to consider clusters with higher idle loads, as the replication process could potentially overload such clusters. Response time differences are minor for this use case and can be considered unimportant. It is relevant when clients don't automatically update their connections based on the response received from the application. In this case clients continue routing through the reverse proxy until the target cluster's IP is manually configured. This results in an overhead since we forward the request not just once per client, but until the IP switch to the target cluster is completed. This configuration directly affects the overall response latency. An availability rate above 98% confirms that the tool is suitable for a migration use case. The duration of the downtime shouldn't be influenced by the size of the DB. The application startup times will most likely affect the downtime because, at boot up, because the target DB must be ready for write operations.

The proposed experiment and the obtained results underlie a few limitations. Firstly, not all types of data were considered during the migration. In particular, no user context (i.e., state of the test application) or session data, such as a Redis store, was included in the test application. The evaluation was also limited to an SQL DB managed by an operator. This only covers a small portion of use cases. This is a proof of concept and we are only showing general applicability. Finally, we also have threats to validity in our experimental setup, for example application startup and network stability, which could impact the results.

7. Conclusion and Future Work

This paper showed an implementation of how a tool for migrating orchestrated environments could be built and an evaluation of this tool. To highlight the contribution of our tool, we conclude that we achieved an almost seamless application migration in edge and fog computing environments with an availability of over 98%. With our tool we are able to establish a secure tunnel between our environments by using Submariner, migrate stateful application by replicating PostgreSQL DBs and forward incoming traffic by rerouting it with a reverse proxy.

While our implementation and evaluation of this tool provided important insights, it also highlighted issues that require further investigation. The evaluation of this tool was performed using single node clusters, which does not reflect reality. In future experiments, it would be interesting to benchmark multi-node clusters to see if the tool's results are comparable. In order to handle traffic forwarding in a more efficient and generic way, different proxy solutions or service meshes that also sound promising, should be investigated in future work. It would also be interesting to look at migrating different DB types such as NoSQL or key-value stores to cover a wider variety of storage methods.

References

- [1] K. Cao, Y. Liu, G. Meng, Q. Sun, An Overview on Edge Computing Research, *IEEE Access* 8 (2020) 85714–85728. URL: <https://ieeexplore.ieee.org/document/9083958/>. doi:10.1109/ACCESS.2020.2991734.
- [2] F. A. Salaht, F. Desprez, A. Lebre, An Overview of Service Placement Problem in Fog and Edge Computing, *ACM Computing Surveys* 53 (2021) 1–35. URL: <https://dl.acm.org/doi/10.1145/3391196>. doi:10.1145/3391196.
- [3] S. Böhm, G. Wirtz, Cloud-Edge Orchestration for Smart Cities: A Review of Kubernetes-based Orchestration Architectures, *EAI Endorsed Transactions on Smart Cities* 6 (2022) e2. URL: <https://publications.eai.eu/index.php/sc/article/view/1197>. doi:10.4108/eetsc.v6i18.1197.
- [4] R. Zheng, J. Xu, X. Wang, M. Liu, J. Zhu, Service placement strategies in mobile edge computing based on an improved genetic algorithm, *Pervasive and Mobile Computing* 105 (2024) 101986. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1574119224001111>. doi:10.1016/j.pmcj.2024.101986.
- [5] C.-H. Hong, B. Varghese, Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms, *ACM Computing Surveys* 52 (2020) 1–37. doi:10.1145/3326066.
- [6] L. Ma, S. Yi, N. Carter, Q. Li, Efficient Live Migration of Edge Services Leveraging Container Layered Storage, *IEEE Transactions on Mobile Computing* 18 (2019) 2020–2033. URL: <https://ieeexplore.ieee.org/document/8470949/>. doi:10.1109/TMC.2018.2871842.
- [7] K. Kaur, F. Guillemin, F. Sailhan, Live migration of containerized microservices between remote Kubernetes Clusters, in: *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, Hoboken, NJ, USA, 2023, pp. 1–6. URL: <https://ieeexplore.ieee.org/document/10225858/>. doi:10.1109/INFOCOMWKSHPS57453.2023.10225858.
- [8] J. Zheng, T. S. E. Ng, K. Sripanidkulchai, Workload-aware live storage migration for clouds, in: *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ACM, Newport Beach California USA, 2011, pp. 133–144. doi:10.1145/1952682.1952700.
- [9] S. Wang, J. Xu, N. Zhang, Y. Liu, A Survey on Service Migration in Mobile Edge Computing, *IEEE Access* 6 (2018) 23511–23528. doi:10.1109/ACCESS.2018.2828102.
- [10] F. Barbarulo, C. Puliafito, A. Viridis, E. Mingozzi, Extending ETSI MEC Towards Stateful Application Relocation Based on Container Migration, in: *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, Belfast, United Kingdom, 2022, pp. 367–376. doi:10.1109/WoWMoM54355.2022.00035.
- [11] Z. Aqun, Y. Yuan, J. Yi, G. Guanqun, Research on tunneling techniques in virtual private networks, in: *WCC 2000 - ICCT 2000. 2000 International Conference on Communication Technology Proceedings (Cat. No.00EX420)*, volume 1, 2000, pp. 691–697 vol.1. doi:10.1109/ICCT.2000.889294.
- [12] IP in IP Tunneling, RFC 1853, 1995. URL: <https://www.rfc-editor.org/info/rfc1853>. doi:10.17487/RFC1853.
- [13] J. Hoagland, S. Krishnan, D. Thaler, Security Concerns with IP Tunneling, RFC 6169, 2011. URL: <https://www.rfc-editor.org/info/rfc6169>. doi:10.17487/RFC6169.
- [14] T. Saad, B. Alawieh, H. T. Mouftah, S. Gulder, Tunneling techniques for end-to-end vpns: generic deployment in an optical testbed environment, *IEEE Communications Magazine* 44 (2006) 124–132.
- [15] U. Bulkan, T. Dagiuklas, M. Iqbal, K. M. S. Huq, A. Al-Dulaimi, J. Rodriguez, On the Load Balancing of Edge Computing Resources for On-Line Video Delivery, *IEEE Access* 6 (2018) 73916–73927. doi:10.1109/ACCESS.2018.2883319.

Comparing Cloud and On-Premises Kubernetes: Insights into Networking and Storage Tooling

Jakob Koller^{1,*}, Sebastian Böhm¹

¹University of Bamberg, An der Weberei 5, Bamberg, 96047, Germany

Abstract

Kubernetes (K8s) is nowadays a well-recognized platform for automating deployment, scaling, and management of containerized workloads. However, running K8s in an on-premises environment comes with unique challenges, because several crucial components are typically managed by the cloud providers. These challenges are especially pronounced in network-specific load balancing services and storage management. For running K8s in an on-premises environment, these functionalities must be provided additionally. To see how open-source tools compare to their cloud counterparts, we conducted experiments evaluating MetalLB and Cilium for networking, as well as Ceph & Rook and Longhorn for storage. The results showed that MetalLB and Cilium, for networking, could achieve similar results to cloud-based K8s. For storage, Ceph & Rook outperformed their cloud counterparts, whereas Longhorn delivered inferior results.

Keywords

Kubernetes, On-premises, Container orchestration, Networking, Storage

1. Introduction

Nowadays, Kubernetes (K8s) is one of the leading platforms for container orchestration. It is widely recognized as the state-of-the-art platform for automating deployment, scaling, and management of containerized workloads [1]. K8s was originally designed with cloud environments in mind, leveraging the scalability and flexibility that cloud services provide [2]. However, there are scenarios where cloud environments may not be ideal. For instance, workloads involving sensitive data that cannot be stored externally often necessitate alternative deployment strategies. In such cases, an on-premises environment becomes viable, enabling organizations to retain full control over their data and systems. Despite its advantages, running K8s on-premises introduces significant challenges. Many cloud providers offering K8s manage several critical components, including storage and networking, as part of their service. In an on-premises setup, these components must be replaced with something equivalent, making the process more complex. Providing equivalents for network and storage on-premises is considered the most significant challenge. Networking includes the load balancer component, which is essential for exposing the deployment to the public. K8s itself does not provide a load balancer; in an on-premises environment, the load balancer component must be replaced with an alternative solution [3]. The cloud environment provides scalable and distributed storage solutions as a managed service for storage. This concept must be adapted for the on-premises environment.

This paper aims to evaluate how tools for networking and storage in an on-premises K8s environment functionally compare to their cloud-based counterparts. To achieve this, we conducted an experiment to address the following research question: **How do networking and storage tools for running K8s in an on-premises environment functionally compare to their cloud-based equivalents?**

To answer our research question, we conduct a reproducible experiment to evaluate whether an on-premises K8s setup, enhanced with additional tooling, can provide a functional experience comparable to a cloud-based environment. By executing the experiment in both environments, our approach ensures

17th Central European Workshop on Services and their Composition (ZEUS), February 20 - February 21, 2025 in Vienna, Austria

*Corresponding author.

✉ jakob.koller@stud.uni-bamberg.de (J. Koller); sebastian.boehm@uni-bamberg.de (S. Böhm)

🌐 <https://www.uni-bamberg.de/en/pi/team/boehm-sebastian/> (S. Böhm)

🆔 0000-0003-3719-1923 (S. Böhm)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

insights into the feasibility, performance, and usability of on-premises solutions compared to their cloud counterparts.

The remainder of the paper is structured as follows: Section 2 discusses the current research on on-premises K8s. Section 3 provides an overview of existing tools that enable K8s functionality in on-premises environments. In Section 4, we leverage these tools to conduct our experiment. Finally, we critically review the experiment in Section 5 and conclude our work in Section 6.

2. Related Work

There are already related approaches to this work that discuss and evaluate solutions for cloud-equivalent on-premises K8s. Packard et al. [4] discussed running and building a K8s cluster in an on-premises environment. They partly discussed the networking components of their cluster, focusing on the external connectivity to the Internet. In addition, they highlighted their solution for the storage aspect. However, they only described a use case-based proof of concept with InfluxDB¹ and K8s. Also, they considered only a small subset of network and storage tools and did not provide a comparison or reasoning for their tool selection. Ruiz et al. [5] used an on-premises K8s cluster to test their custom and QoS-aware autoscaling of deployments in different cluster setups. They provided a custom load balancer to address network challenges and did not use publicly available open-source solutions. Tackling storage-related aspects for on-premises K8s was not in scope. Manaouil and Lebre [6] discussed K8s in the domain of edge computing, especially the applicability of geographically distributed K8s clusters. They used a basic cluster setup not designed for production usage for testing. Mondal et al. [7] set up a K8s cluster from scratch without any auxiliary tooling. The deployments were only internally exposed. Hence, a solution for load balancing was not discussed. Böhm and Wirtz [8] compared different K8s distributions based on their performance characteristics. However, only the baseline functionality is included and needs to be extended for a production-like environment.

None of the related works discussed a cloud-equivalent on-premises setup, considering both network and storage aspects. Specifically, no empirical and quantitative evaluation yet shows the outcome when K8s nodes fail. Consequently, this work wants to address these gaps by providing an overview of tooling with their functional comparability to cloud-based K8s.

3. Tooling

This section discusses the available tooling for networking and storage. To identify an eligible set of tools for evaluation, we first followed the findings mentioned in the previously discussed related work (Section 2). Additionally, we enriched the set of tools by researching the internet for further solutions. We eliminated solutions that do not contribute to our goal of cloud equivalence from a feature perspective. Furthermore, we do not consider tools with minor reputations, incomplete documentation, or inactive development.

3.1. Networking

Our research revealed two classes of solutions exist for providing load balancing for services, particularly load balancers and via the Container Network Interface (CNI). We selected two representative tools for the identified categories: MetalLB as the load balancer and Cilium as the CNI.

MetalLB. One of the most mature and widely used load balancers for on-premises K8s environments is MetalLB [9]. It operates in Layer 2 mode or Border Gateway Protocol (BGP)² mode. In Layer 2 mode, MetalLB uses the Address Resolution Protocol (ARP) and Neighbor Discovery Protocol (NDP) to assign multiple IP addresses to a single machine. However, this mode has one significant limitation: the

¹<https://www.influxdata.com/>

²BGP is a routing protocol used to exchange network reachability information between systems, enabling efficient traffic distribution and scalability in complex network environments [10].

incoming traffic from outside the cluster is limited by the bandwidth of a single node. All incoming traffic must pass through this node before being distributed across the cluster, creating a potential bottleneck. The second mode uses BGP to establish a direct BGP peering session between the router and the different nodes. This allows a BGP-compliant router to forward traffic directly to the appropriate node, bypassing the single node bottleneck³.

Cilium. Unlike MetalLB, which is solely a load balancer, Cilium is primarily a CNI with additional features. One of the features is the capability to provide load balancing for services. Similar to MetalLB, it supports both Layer 2 and BGP mode. Since K8s requires a CNI by design, using the integrated load balancer from Cilium eliminates the need to install additional tools to provide load balancing⁴.

3.2. Storage

Longhorn. Developed by Rancher, Longhorn is an open-source block storage system. It claims to be lightweight and reliable, which could be important in resource-constrained environments. It provides incremental snapshots of the block storage, automated backups to secondary storage, replication of block storage across multiple nodes or even data centers, non-disruptive upgrades, and an intuitive dashboard⁵.

Ceph & Rook. Rook allows Ceph⁶ storage to be used natively in K8s. Ceph is a highly scalable distributed storage solution for block storage, object storage, and shared file systems. Rook is a framework that automates the deployment and management of Ceph to provide self-managing, self-scaling, and self-healing storage. Rook achieves this by using K8s resources to deploy, configure, provision, upgrade, and monitor Ceph. Like Longhorn, Ceph can automatically replicate data to other available nodes to protect the cluster from data loss⁷.

4. Functional Comparison

This chapter presents the functional comparison between the cloud-based and on-premises K8s setups. We describe the experimental environment and procedure and present, analyze, and discuss the results.

4.1. Experimental Environment

The test environment for the on-premises setup consists of five locally hosted Virtual Machines (VMs) distributed across multiple machines. Each VM is equipped with 2 vCPUs, 4 GB of RAM, a 40 GB SSD boot disk, and an additional 12 GB disk for the storage experiment. These five VMs are configured as a highly available K8s cluster using K3s⁸ as the K8s distribution and Cilium⁹ as the base CNI.

We use DigitalOcean's managed K8s Service for the cloud environment. The cluster is configured with the same specifications as the on-premises setup. For storage, the managed cluster automatically uses DigitalOcean's managed Block Storage service. The experiment is performed 5 times for each tool to evade any potential coincidences.

4.2. Network Experiment

The goal of the network experiment is to evaluate the load-balancing components of the cluster. In the following, we describe our experiment design and architecture. Afterward, we present our results, revealed by the experiment.

³<https://metallb.io/>

⁴<https://docs.cilium.io/en/stable/network/lb-ipam/#services>

⁵<https://longhorn.io/docs/1.7.2/>

⁶<https://ceph.io/>

⁷<https://rook.io/docs/rook/latest/Getting-Started/intro/>

⁸<https://k3s.io/>

⁹<https://cilium.io/>

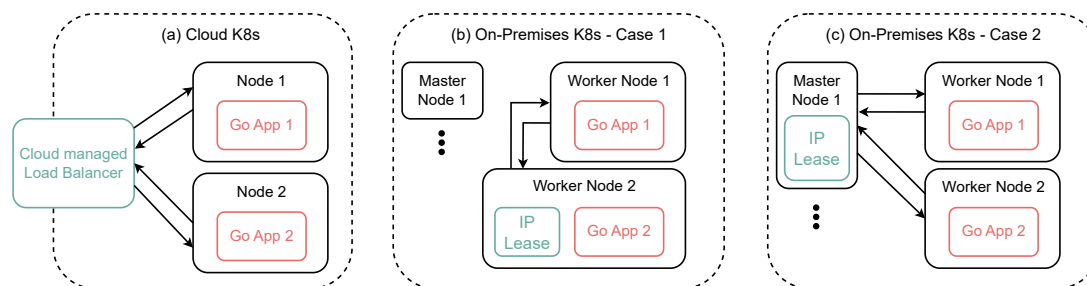


Figure 1: Experimental Design and Approach of the Network Test.

4.2.1. Network Experiment Design

The experiment architecture involves a simple client-server interaction¹⁰. As seen in Figure 1, a Golang-based HTTP server application is deployed in the cluster, with two replicas running on separate worker nodes. The worker exposes an endpoint that is routed out of the cluster using an ingress controller exposed via the IP address provided by the load balancer. A client application from a different network sends an HTTP GET requests to the server’s endpoint every 100 milliseconds and logs the response. If there is a downtime detected, the application will measure the downtime and count the failed requests.

The cloud K8s cluster, as illustrated in Figure 1 describes the experiment architecture of the cloud experiment. In this setup, we simulate a node failure by shutting down a node containing the workload while logging the client’s response. The managed load balancer should ensure that the is dynamically redistributed to the healthy node. For the on-premises environment, we need to slightly change the experiment architecture, because the tools are configured using Layer 2 mode. As already highlighted in Section 3, in Layer 2 mode, only one node in the cluster holds the lease for the service IP address at any given time. If traffic arrives on the node the holds the current lease, it then gets forwarded to the appropriate Pod running the workload. This fundamental difference in traffic routing introduces challenges for direct comparisons with the cloud setup. To address these challenges and ensure a fair evaluation, we split the experiment for the on-premises environment into two distinct cases:

1. **Worker Node Holding the Lease:** As illustrated in the first case in Figure 1, the worker node holds the service IP lease and hosts one of the workloads. When this node fails, the lease for the IP and the workload are interrupted, which is closer to the experiment performed in the cloud environment. However, it isn’t fully representative, since the IP lease doesn’t necessarily have to be announced by the worker node.
2. **Master Node Holding the Lease:** In the second case in Figure 1, the master node holds the service IP lease. Simulating a node failure on the master would require the load balancer to elect a new node to announce the IP lease. However, comparing this to the cloud environment would be unfair, as we don’t interrupt one of the workloads.

For reference, we also tested the case where we simulated the failure on the worker node while the master held the lease. In combination with the other two cases, this will help us to precisely define the downtime that is created by the load balancer.

4.2.2. Network Experiment Results

For the cloud environment, the client reported an average of 2.60 ($\sigma = 0.89$) failed requests, resulting in a downtime of 2.77 seconds ($\sigma = 0.57$). A small amount of failed requests have to be expected, since there are requests being sent as the failed node was going offline.

In the on-premises environment, we can see that both Cilium and MetalLB returned similar results for the first case of the experiment, where we simulate a node failure on the worker node with a workload and the IP lease. Both MetalLB and Cilium had an average of ≈ 75 failed requests. If we compare this to

¹⁰<https://github.com/jacolate/KubernetesTesting>

Table 1

Results for the Network Experiment (μ/σ), values have been averaged over five runs of the experiment

Case	Fig. 1	Failure Simulated	Current Lease	Total Requests	Failed Requests	Downtime (sec.)
Reference	-	Worker	Master	800	74.40/2.30	16.48/0.52
Cloud	(a)	-	-	800	2.60/0.89	2.77/0.57
Cilium	(b)	Worker	Worker	800	74.40/6.69	19.56/2.44
	(c)	Master	Master	800	12.40/13.88	6.15/6.61
MetalLB	(b)	Worker	Worker	800	74.60/6.38	17.33/1.46
	(c)	Master	Master	800	0.00/0.00	0.00/0.00

our reference case, where we have an average of 74.40 ($\sigma = 2.30$) failed requests, we can see that these values are similar. This shows that created downtime doesn't necessarily come from the load balancer, but from other cluster internals. The real impact of the load balancer on the failed requests can be observed in the second case of the experiment. Here MetalLB showed no failed requests at all, whereas Cilium showed an average of 12.40 failed requests, but with a high standard deviation of 13.88.

4.3. Storage Experiment

Similar to the network experiment, we designed an experiment to evaluate the performance and reliability of the storage tooling. The following section outlines the design and architecture of the storage experiment, followed by a presentation and analysis of the results obtained.

4.3.1. Storage Experiment Design

To evaluate the different tools for supporting distributed storage in K8s, we developed a custom K8s workload consisting of an application written in Golang and a MySQL database. At the start of the experiment, the application establishes a connection to the database and writes a unique character sequence to it (Figure 2). Afterward, the application enters a loop, validating the character sequence every 100 milliseconds.

The experiment is divided into two scenarios. In the first scenario, the workload is simply rescheduled to a different node. In the second scenario, a simulated node failure is performed by manually shutting down a node. Both scenarios allow us to measure how long it takes for the Golang application to re-establish a connection to the database. We also verify whether the previously written data remains intact after each scenario.

4.3.2. Storage Experiment Results

In the cloud environment, the first scenario, involving rescheduling the MySQL deployment, resulted in the Golang application losing its connection to the database for an average of 7.42 seconds ($\sigma = 0.40$).

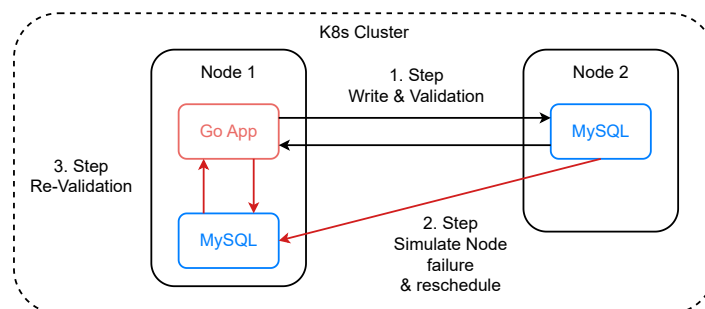


Figure 2: Experimental Design and Approach of the Storage Test.

However, once the connection was re-established, the application successfully validated the data in the database. The second scenario, simulating a node failure, encountered some issues. After the node crash, K8s automatically rescheduled the deployment to another node. However, the Pod failed to start and remained stuck in the *Creating* phase. The root cause was that the Block Storage volume was still attached to the failed node, preventing it from being remounted to the new node.

For on-premises, the results of the first scenario were similar to the cloud setup. With Rook & Ceph installed as a storage provider, the Golang application lost its connection for an average of 4.02 seconds ($\sigma = 0.40$). In contrast, using Longhorn, took significantly longer, with an average downtime of 20.08 seconds ($\sigma = 1.63$). The second scenario in the on-premises environment produced results similar to those observed in the cloud. After the node failure, the Pod could not be started because the volume remained attached to the failed node, preventing it from being remounted. This issue was consistent across both Rook & Ceph and Longhorn.

5. Discussion

The evaluation of the network tooling showed that the load balancing capabilities of an on-premises environment are similar to the cloud environment. However, the results also showed cases where we have an increased number of failed requests in the on-premises environment compared to the cloud environment. As highlighted, most of these failed requests aren't coming from the load balancer but from other cluster internals, which require further investigation.

For storage, the experiment showed, that Rook & Ceph can achieve similar results as their cloud counterpart. Longhorn however, performed worse, with significantly longer recovery times. The reason for this discrepancy requires further investigation. In the second scenario, all tested tools—including Rook & Ceph, Longhorn, and the cloud environment experienced the same issue: the inability to remount the storage volume to a new node due to its attachment to the failed node. Notably, Longhorn's documentation acknowledges this as expected behavior and requires administrative intervention¹¹.

The proposed experiment and the results underline a few limitations. First, the selection of tools was limited to the most popular ones. Numerous alternative tools exist, each with potentially distinct capabilities, limitations, and performance characteristics. Secondly, the network tools were only tested in the Layer 2 configuration. Using the BGP configuration instead should, in theory, be preferable. Furthermore, the comparison between cloud and on-premise tooling was based on selected functional experiments. A more comprehensive benchmark may provide more detailed results.

6. Conclusion and Future Work

This paper presented an approach to compare on-premises and cloud-based K8s environments, specifically focusing on network and storage tooling. To answer our research question, we conclude that it is possible to achieve comparable functionality in on-premises environments in the area of storage and networking using appropriate tools. MetalLB and Ceph & Rook delivered comparable results to the cloud environment, demonstrating similar performance in terms of load balancing and storage functionality. While our research provided valuable insights, there is still room for improvement. For our future work, we want to expand the number of tools tested for a more comprehensive evaluation. Additionally, testing network tools in BGP mode over Layer 2 mode, could uncover a performance increase. Finally, expanding the scope to include other areas of K8s, such as monitoring, automated deployment, and authentication, would offer a more comprehensive understanding of the trade-offs between cloud-based and on-premises K8s setups.

¹¹<https://longhorn.io/docs/1.7.2/high-availability/node-failure/>

References

- [1] M. A. Rodriguez, R. Buyya, Container-based cluster orchestration systems: A taxonomy and future directions, *Software: Practice and Experience* 49 (2019) 698–719. URL: <https://onlinelibrary.wiley.com/doi/10.1002/spe.2660>. doi:10.1002/spe.2660.
- [2] P. Kayal, Kubernetes in Fog Computing: Feasibility Demonstration, Limitations and Improvement Scope : Invited Paper, in: *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, IEEE, New Orleans, LA, USA, 2020, pp. 1–6. doi:10.1109/WF-IoT48130.2020.9221340.
- [3] K. Takahashi, K. Aida, T. Tanjo, J. Sun, A Portable Load Balancer for Kubernetes Cluster, in: *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, ACM, Chiyoda Tokyo Japan, 2018, pp. 222–231. URL: <https://dl.acm.org/doi/10.1145/3149457.3149473>. doi:10.1145/3149457.3149473.
- [4] M. Packard, J. Stubbs, J. Drake, C. Garcia, Real-World, Self-Hosted Kubernetes Experience, in: *Practice and Experience in Advanced Research Computing*, ACM, Boston MA USA, 2021, pp. 1–5. URL: <https://dl.acm.org/doi/10.1145/3437359.3465603>. doi:10.1145/3437359.3465603.
- [5] L. M. Ruiz, P. P. Pueyo, J. Mateo-Fornes, J. V. Mayoral, F. S. Tehas, Autoscaling Pods on an On-Premise Kubernetes Infrastructure QoS-Aware, *IEEE Access* 10 (2022) 33083–33094. URL: <https://ieeexplore.ieee.org/document/9732997/>. doi:10.1109/ACCESS.2022.3158743.
- [6] K. Manaouil, A. Lebre, Kubernetes and the Edge?, PhD Thesis, Inria Rennes-Bretagne Atlantique, 2020.
- [7] S. K. Mondal, R. Pan, H. M. D. Kabir, T. Tian, H.-N. Dai, Kubernetes in IT administration and serverless computing: An empirical study and research challenges, *The Journal of Supercomputing* 78 (2022) 2937–2987. URL: <https://link.springer.com/10.1007/s11227-021-03982-3>. doi:10.1007/s11227-021-03982-3.
- [8] S. Böhm, G. Wirtz, Profiling Lightweight Container Platforms: MicroK8s and K3s in Comparison to Kubernetes, 2021.
- [9] B. Johansson, M. Ragberger, T. Nolte, A. V. Papadopoulos, Kubernetes Orchestration of High Availability Distributed Control Systems, in: *2022 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, Shanghai, China, 2022, pp. 1–8. doi:10.1109/ICIT48603.2022.10002757.
- [10] Y. Rekhter, S. Hares, T. Li, A Border Gateway Protocol 4 (BGP-4), RFC 4271, 2006. URL: <https://www.rfc-editor.org/info/rfc4271>. doi:10.17487/RFC4271.

Object Instance Monitoring

Lisa Arnold¹

¹*Institute of Databases and Information Systems, Ulm University, Germany*

Abstract

The monitoring of business processes represents a pivotal component of contemporary management practices, with the objective being the assurance of the efficiency and effectiveness of company processes. Object-centric business process monitoring constitutes a concept that involves the continuous observation and analysis of object instances, with the purpose of identifying deviations from defined standards and the initiation of prompt measures for optimisation. The utilisation of specific technologies, such as Object Instance Monitoring, empowers companies to acquire valuable insights that facilitate the enhancement of performance and the augmentation of customer satisfaction. The ability to identify risks at an early stage, optimise the use of resources, and increase the company's agility are key to long-term success in an increasingly dynamic business world. The objective of this paper is twofold. Firstly, it seeks to define process metrics (i.e. the status and duration of instances). Secondly, it discusses visualisation techniques for object instance monitoring. The purpose of these two objectives is to facilitate resource and risk management.

Keywords

process monitoring, business processes, object-centric, object instance monitoring, resource management

1. Introduction


Object instance monitoring (OIM) is an essential aspect of process management. It facilitates the real-time tracking and analysis of individual object instances, enabling companies to make informed decisions. An object's instance (e.g., Instance *Application1*) is defined as a specific execution of a business object (e.g., Object *Application*). This is characterised by various connected states with business attributes and decisions (i.e., lifecycle processes). The primary objectives of monitoring are to ensure process quality, identify bottlenecks and increase efficiency. The implementation of individual process monitoring, for instance through the utilisation of dashboards, empowers organisations to respond expeditiously to deviations and proactively implement measures [1].


Traditional business processes are defined as a series of activities and order constraints. These systems offer the user a process-centric perspective. The monitoring of such traditional business processes has the potential to provide Business Activity Monitoring (BAM) [2, 3]. However, information regarding the specific execution of individual activity instances is not possible. Consequently, the execution of these activities occurs within a black box, which is inaccessible to the user [4].

16th Central European Workshop on Services and their Composition, February 20 – 21, Vienna, Austria

✉ lisa.arnold@uni-ulm.de (L. Arnold)

ORCID  0000-0002-2358-2571 (L. Arnold)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

The PHILharmonicFlows framework is a runtime engine that facilitates the execution of object-centric business processes. Furthermore, a monitoring engine that is integrated within the runtime engine is provided. The fundamental premise of this paper is to define the concept of object instance monitoring, which will be integrated within the monitoring engine. In order to facilitate the monitoring of object-centric business processes, it is essential that each status (e.g. running or terminated) of an instance that may be in existence during runtime is defined and delimited. In addition to the monitoring of status, the duration of each instance constitutes a fundamental element of OIM. The objective of this paper is to establish metrics to measure *On Time*, *On Risk*, and *Overdue* instances. In addition, the utilisation of visualisation techniques to illustrate OIM is discussed. In light of this, a range of visualisation techniques is hereby proposed, adapted to object-oriented processes and their respective instances.

The remainder of this paper is structured as follows. Section 2 provides a concise overview of the key principles and characteristics of object-centric business processes. In Section 3, an overview is provided of the potential statuses of instances during runtime. The concept and idea of instance duration monitoring is defined in Section 4. Section 5 provides a detailed exposition of the visualisation techniques that have been developed for the purpose of object instance monitoring. Related work is discussed in Section 6. Section 7 concludes the paper.

2. Fundamentals of object-centric Business Process

In the object-centric process management paradigm PHILharmonicFlows, a business process is described in terms of interacting business objects that correspond to real-world entities. The interactions between the objects, as well as their relations, including their cardinalities, hierarchical structure, and semantic relations, are manifested in the **Relational Process Structure** (RPS) (cf. Fig. 1)[5]. During execution, it is possible for business objects to create any number of object instances, provided that the constraints imposed by their cardinalities are respected. Furthermore, business attributes may be defined for each business object, specifying the business process. The RPS corresponding to the recruitment business process, along with a number of its business attributes, is depicted in Fig. 1.

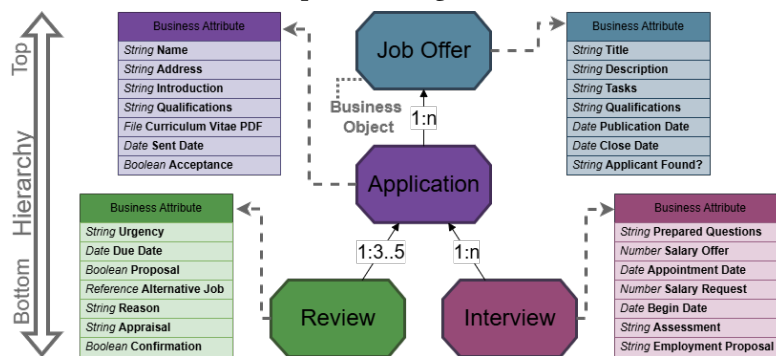


Figure 1: RPS of the recruitment business process with a number of the objects' business attributes.

The runtime behaviour of these business objects is defined in terms of **object lifecycles** (lifecycle for short) [6]. The lifecycle of the object *Job Offer* is depicted in Fig. 2. In general, a lifecycle comprises of *states*, with one start state (*Preparation*) and at least one end state (*Position*

Filled and *Position Vacant*), as well as any arbitrary number of intermediate states (*Published* and *Closed*). The runtime behaviour of each object instance is defined by its own lifecycle instance. The start state of an instance is automatically assigned as *activated* when the instance is created. Moreover, it is important to note that during the execution of an instance, it is only permissible for one state of the lifecycle to be marked as *activated* at any given moment. It is consequently evident that parallel execution within a single lifecycle process is not a possibility. However, the execution of multiple instances in parallel is indeed feasible. In order to facilitate user interaction at runtime for each state, an automatically generated *form sheet* is built from the lifecycle structure. In particular, the states of a lifecycle define the form sheets and their steps, which in turn design the input fields. These are constructed from the object attributes. The result is a data-driven business process. Furthermore, the lifecycle may encompass backward transitions to previous states, with the objective of reading and verifying or adjusting previously entered data. However, it should be noted that by default, there are no backward transitions, as these must be explicitly set by a modeller. Moreover, the intention behind backward transitions is not to establish loops, wherein a new instance is generated. Instead, the previous form is exhibited once more, accompanied by the input variables that have previously been entered by an end user.

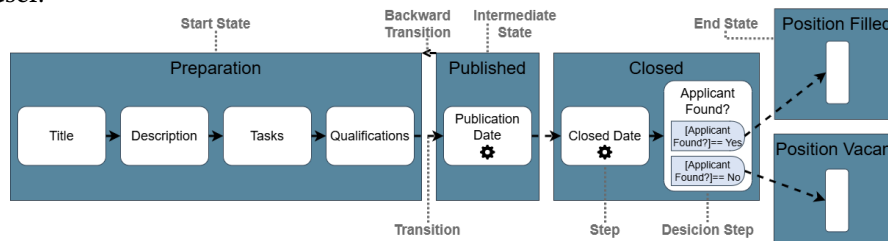


Figure 2: Lifecycle of the object *Job Offer*.

A **coordination process** (cf. Fig. 3) controls the interactions between the lifecycles of multiple objects and defines the sequence of states between multiple lifecycle states. A coordination step is defined as a reference to the lifecycle state of an object. Moreover, a coordination process is generally represented by a graph, in which the vertices correspond to the coordination steps and the edges correspond to the coordination transitions. The coordination process graph is defined as a directed, acyclic and connected graph. This implies that it does not permit backward transitions or loops to preceding coordination steps [7]. Conversely, the absence of such mechanisms can engender cyclic dependencies, which, in turn, can precipitate deadlocks. Consequently, the occurrence of cyclic dependencies may result in deadlocks, thereby introducing an inherent risk to the process [7].

3. Status of an Instance

This section provides an overview of the various statuses that can be assigned to an instance.

Running Instance: A running instance is defined as an instance with an activated state that is not the end state. In Fig. 4 the start state *Preparation* is the active state. Data collection is done using forms. The form can be automatically generated from the process elements. Normally, in activity-centric processes, forms have to be designed manually for each activity, but since

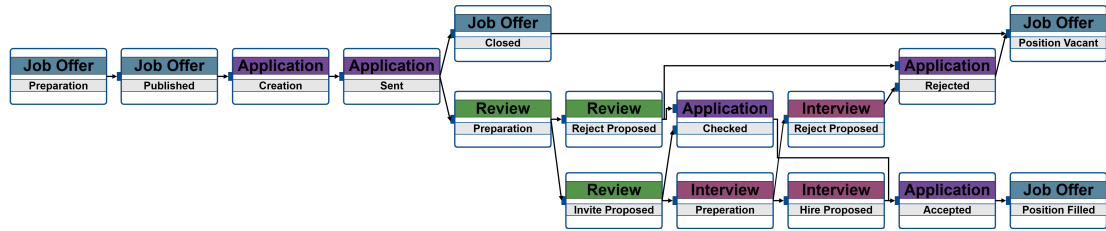


Figure 3: The coordination process of the coordinating business object *Job Offer*.

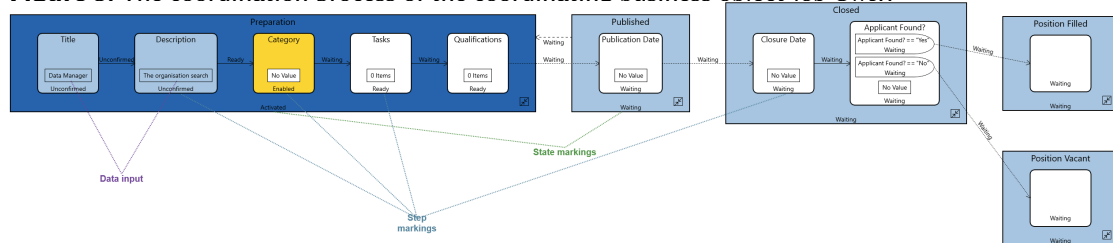
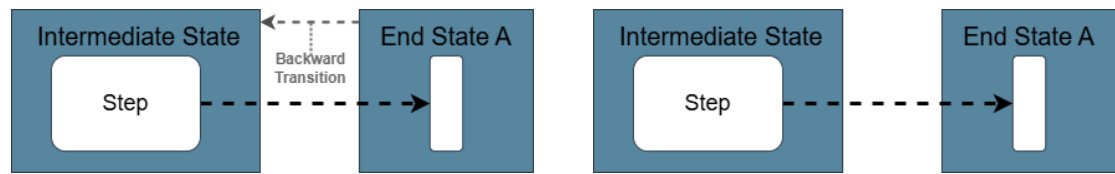


Figure 4: Screenshot from runtime engine of the Lifecycle *Job Offer* with markings.

PHILharmonicFlows is data-centric, form generation is built in. The operational semantics control the dynamic aspects of the form, such as the next value that is required. The active state of the process (cf. Fig. 4) gives rise to the current form sheet *Preparation*, incorporating the designated input fields, which include *Title*, *Description*, *Category*, *Tasks*, and *Qualifications*. Input fields designated *Title* and *Description* are filled in with data, whereas a value for *Category* is required, indicated by the marking *Enabled* on the *Category* step (cf. Fig. 4). The input fields designated *Tasks* and *Qualifications* have been marked as *ready*. Nevertheless, the operational semantics have been designed to allow for flexibility. It is not obligatory for a user to complete the form in the correct order. However, they are at liberty to deviate from this by filling in the *Qualifications* field first and the *Tasks* field afterwards. This provides the end user with a certain degree of flexibility. Provided that a lifecycle instance is active (not marking the end state as *active*), it is considered to be in a running state.

Terminated Instance: Terminated instances are defined as instances that have reached one of their end states (end state marked as *activated*). As demonstrated in Fig. 2 and 4, the end states comprise precisely one step. This step is devoid of any reference to a business attribute and is consequently designated as an *empty step*. It is important to note that an end state does not generate a form sheet or input fields. The fundamental purpose of end states is to enable the monitoring or collection of terminated instances without the necessity of deleting or eliminating these instances. A lifecycle can be characterised by multiple end states. The various potential end states of each instance (e.g. the state of a position being either vacant or filled) can be leveraged in order to demonstrate their different outcomes. In general, these instances do not engender further work. End states can exhibit backward transitions to one of their previous states. It is incumbent upon the modeller to define such transitions explicitly. In the event of such a backward transition, a terminated instance may be reactivated. This enables the distinction between instances that have been *fully* and *non-fully* terminated.

Fully Terminated Instance: A fully terminated instance is defined as an instance that has reached its end state and there is no backward transition to reactivate the instance again. Fig. 5b



(a) Backward transition from an end state to a previous state.

(b) No outgoing Backward transition from an end state.

Figure 5: Different between completely terminated instances and temporary terminated instances.

shows an intermediate state with one step and a transition to the empty end state. An instance that can be fully terminated is the default way of modelling a lifecycle process.

Non-Fully Terminated Instance: A non-fully terminated instance is defined as an instance that has reached its end state and there is at least one backward transition to a previous state modelled to reactivate the instance. Fig. 5a shows the same constellation as in Fig. 5b with an additional backward transition. This case is used when an instance that has reached an end state and is normally terminated can be considered again. Considering the recruitment process described in Section 2, one use case is the rejection of applications because the vacancy is filled, as illustrated in Figure 2. However, the candidate who has been offered the job rejects it. In this case, the rejected application is reconsidered and the instance to find the most suitable candidate is reactivated. Furthermore, it is possible for a lifecycle to have two end states, one with the possibility of reactivating an instance and another end state with no reactivation option. This avoids the need for complex special case modelling.

Expected Instance: Expected instances are defined as those which are yet to be created, but whose existence is already anticipated. The purpose of these instances is to facilitate resource management calculations and planning.

- **Minimum Cardinality:** Within the paradigm of the data model, the cardinality between two objects that are related can be defined. By default, a $1 : n$ cardinality is established, and the $n : m$ relation's placeholders n and m can be defined by the modeller as follows: minimum (e.g., $4..m$), maximum (e.g., $1..8$), or range (e.g., $3..5$). In the event of a minimum being specified, it can be used to calculate the expected instances. To illustrate this, consider the recruitment process illustrated in Fig. 1. This process stipulates that for each application, between three and five instances of reviews must be created.
- **Event Log:** The utilisation of an event log facilitates the calculation of the expected number of instances by leveraging the average number of instances generated by completed business process instances. However, it should be noted that this variation can be subject to inaccuracy. Nevertheless, it can assist in approximating the anticipated expense. For instance, the number of applications can be estimated during the recruitment process. Consequently, the anticipated quantity of application instances can be calculated, thereby facilitating the estimation of the number of review instances.
- **Machine Learning:** The utilisation of supervised machine learning (i.e., neural networks [8]), in conjunction with a comprehensive event log, has been demonstrated to enhance the precision of the mean calculation of the aforementioned method (i.e., event log). This approach has been evidenced to engender more accurate predictions, as it incorporates

additional factors in addition to the number of applications. The factors that have been taken into consideration are manifold. These include the nature of the job offer, the qualifications that are deemed to be necessary, and economic as well as environmental factors.

Deleted Instances: The PHILharmonicFlows framework facilitates the direct deletion of running instances by end users, contingent upon the possession of the requisite permissions. For instance, an applicant has the option to withdraw their application by deleting it. It is not possible for an employee to remove an application in the absence of permissions.

4. Instance Duration Monitoring

The maximum processing time (i.e. the time span in which the instance must be completed) and the proceeding time (i.e. the time taken to complete the work of an instance) can be used to monitor risk management. For each lifecycle, a modeller (or process owner) can specify the amount of time for an instance, as well as each of its states, to be completed. For example, after an application has been received, a reviewer has 2 weeks to complete the review. Additionally, the maximum processing duration of one lifecycle state (i.e., one form sheet) can be defined. For example, if a candidate is successful in the assessments, they will be invited for an interview (i.e., state *Preparation Interview*). In this case, a recruiter has 5 days to send the candidate an interview date. Each running instance can be categorised into one of the following modes:

On Risk: This is defined for instances or states where the proceeding time is nearly over. The determination of the point at which an instance or state is deemed to be at risk is made by a key performance indicator (KPI). By default, the value *On Risk* is defined as 80%. Consequently, when 80% of the defined proceeding time has elapsed, an instance or state is flagged as being at risk. In the event of an instance or state attaining a status of risk, an alert is to be dispatched to the employees responsible for editing the instance or state. Furthermore, in the event of a reported absence (e.g. illness, holiday), the alert is to be forwarded to a colleague. However, this is only possible if the employee has designated a colleague in the system.

On Time: The definition is applicable to instances or states in which the proceeding time is on time, i.e. the proceeding time falls within the first 80% of the time span. It is possible to adapt the KPI of the value *On Risk*. Consequently, the temporal span within which an instance or state is considered to be on time is also subject to adaptation.

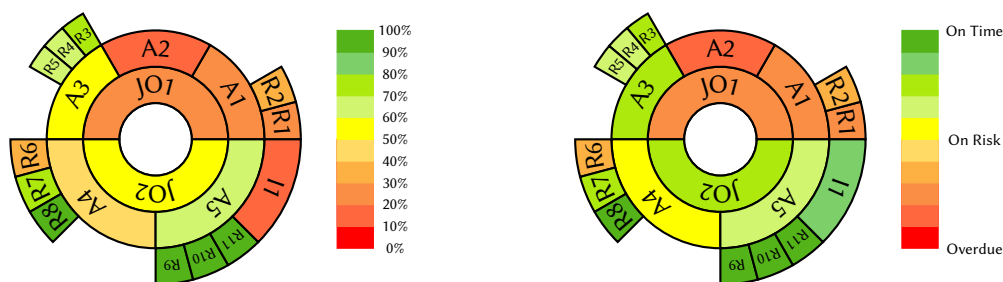
Overdue: This is defined for instances or states where the defined processing time has elapsed. In such a scenario, an escalation message will be transmitted to a different employee. The identification of the employee concerned is specified within the runtime engine. Typically, the escalation message is sent to a colleague the first time, or to a manager if the deadline is missed more often.

5. Visualisation

In the context of object-centric business processes, the PHILharmonicFlows monitoring framework has been developed to facilitate the monitoring of instances created within its runtime

engine. The monitoring framework enables users to define bespoke monitoring components and charts, thereby empowering them to meticulously track and analyse process metrics. The fundamental objective of the monitoring framework is to facilitate a collaborative process in which users can construct their own monitoring framework through a straightforward click-and-build interface. This will enable users to create bespoke monitoring dashboards in accordance with their respective permissions, specific task area and individual interests. The monitoring framework provides a range of options for data visualisation, including pie charts, doughnut charts, and bar charts. These visualisation charts can be employed to represent the instance duration (i.e., *On Time*, *On Risk*, or *Overdue*) or status (i.e., running, terminated, or expected) of a single instance. Furthermore, the framework provides options for data visualisation at the level of a collection of instances (e.g., all application instances related to the same job offer), all instances of the same object type, or the lifecycle state of an instance.

The employment of rudimentary diagrams (e.g. bar charts) is optimal for the monitoring of individual instances (i.e. single review instances) or instances of the same type (i.e. all review instances). The utilisation of sophisticated diagram types is imperative for the comprehensive monitoring of business processes. The sunburst chart [9] is a prime example of a sophisticated diagram. It is particularly well-suited to the visualisation of hierarchically dependent data structures. The adaptation of sunburst charts for the illustration of our recruitment process example is demonstrated in Fig. 6. The sunburst chart [10] was applied to the progress indicator (cf. Fig. 6a) calculated with a one-dimensional *Kalman Filter* [11] and the risk management (cf. Fig. 6b) of all running instances.



(a) Current progress (from 0% to 100%). (b) Current status (*On Time*, *On Risk*, or *Overdue*).
Figure 6: Sunburst charts illustrate the runtime behaviour of the recruitment business process (JO := *Job Offer*, A := *Application*, R := *Review*, I := *Interview*) [10].

Heat maps [1] are a further method of gaining an overall view of the business process. The coordination process is visualised through the utilisation of a heat map. Two variants of the coordination process are distinguished: the standard coordination process (cf. Fig. 3), which displays solely the coordination steps that have interactions with other objects, and the extended coordination process, in which all absent coordination steps (i.e., coordination steps without interactions between other objects) are automatically incorporated. The employment of a heat map facilitates the representation of the number of active instances for each coordination step. In this representation, the colour red is used to denote a high number of active instances, whereas green is used to denote a low number of active instances in a coordination step. The heat map has the potential to optimise resource management by helping to identify personal bottlenecks more efficiently and to take prompt countermeasures.

6. Related Work

The field of Business Activity Monitoring (BAM) has already been established as a subject of research, with a considerable number of academic papers [2, 3, 12]. Furthermore, BAM has been incorporated into commercial tools, such as Bizagi [13]. The Business Activity Monitoring tool from Bizagi is an analytical instrument that enables the graphical representation of information pertaining to the status of ongoing cases. Business Activity Monitoring (BAM) is comprised of three constituent elements. Primarily, *Process BAM* is responsible for the analysis of the present status (i.e. *On Time*, *On Risk*, or *Overdue*) of all ongoing processes. Secondly, the function of *Activity BAM* is to analyse the current state (i.e. *On Time*, *On Risk*, or *Overdue*) of ongoing activities. Finally, the function of the *Resources Monitor* is to analyse the current workload (i.e. *On Time*, *On Risk*, or *Overdue*) and performance of end users and work teams[14]. Furthermore, Camunda employs the heat map to facilitate the monitoring and optimisation of business processes, with the objective of visualising the duration or most frequency path of these processes [1].

7. Conclusions

The objective of this paper is to examine the application of object instance monitoring (OIM) in the context of object-centric business processes. The paper establishes and delineates a comprehensive categorisation of the statuses (e.g., *running* or *terminated*) that may be achieved by an instance during the execution of a business process. Furthermore, it establishes a duration concept for running instances, and specifies metrics *On Time*, *On Risk*, and *Overdue*. Finally, the utilisation of visualisation techniques (i.e. rudimentary diagrams, sunburst charts, and heat maps) adapted to object-centric business processes to illustrate OIM constituents is explained. Further work is currently underway to develop personalised dashboards, the functionality of which will enable each user to create their own bespoke dashboard by selecting a combination of monitoring functions and visualisation types from a curated list.

Acknowledgments. *This work is part of the ProcMape project, funded by the KMU Innovativ Program of the Federal Ministry of Education and Research, Germany (F.No. 01IS23045B).*

References

- [1] E. Amann, C. Corea, C. Drodts, P. Delfmann, A dashboard creator suite for simultaneous predictive process monitoring., in: BPM (PhD/Demos), 2022, pp. 152–156.
- [2] J.-P. Friedenstab, C. Janiesch, M. Matzner, O. Muller, Extending bpmn for business activity monitoring, in: 2012 45th Hawaii International Conference on System Sciences, IEEE, 2012, pp. 4158–4167.
- [3] W. Schmidt, Business activity monitoring (bam), in: Business Intelligence and Performance Management: Theory, systems and industrial applications, Springer, 2013, pp. 229–242.
- [4] V. Künzle, M. Reichert, Philharmonicflows: towards a framework for object-aware process

- management, *Journal of Software Maintenance and Evolution: Research and Practice* 23 (2011) 205–244.
- [5] S. Steinau, K. Andrews, M. Reichert, The relational process structure, in: *Int. Conf. on Advanced Information Systems Engineering (CAiSE'18)*, Springer, 2018, pp. 53–67.
 - [6] S. Steinau, K. Andrews, M. Reichert, Executing lifecycle processes in object-aware process management, in: *Int. Symp. on Data-Driven Process Discovery and Analysis (SIMPDA'17)*, Springer, 2017, pp. 25–44.
 - [7] S. Steinau, K. Andrews, M. Reichert, Coordinating large distributed relational process structures, *Software and Systems Modeling* 20 (2021) 1403–1435.
 - [8] H. Liu, M. Xu, Z. Yu, V. Corvinelli, C. Zuzarte, Cardinality estimation using neural networks, in: *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, 2015, pp. 53–59.
 - [9] F. Le Guen, Sunburst chart, 2022. URL: <https://www.excel-exercise.com/sunburst-chart/>.
 - [10] L. Arnold, M. Breitmayer, M. Reichert, Monitoring object-centric business processes: an empirical study, in: *International Conference on Research Challenges in Information Science*, Springer, 2023, pp. 327–342.
 - [11] L. Arnold, M. Breitmayer, M. Reichert, A one-dimensional kalman filter for real-time progress prediction in object lifecycle processes, in: *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)*, IEEE, 2021, pp. 176–185.
 - [12] H. Kim, Y.-H. Lee, H. Yim, N. W. Cho, Design and implementation of a personalized business activity monitoring system, in: *Human-Computer Interaction. HCI Applications and Services: 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part IV 12*, Springer, 2007, pp. 581–590.
 - [13] F. M. Nafie, M. A. Eltahir, Real-time monitoring and analyzing business process performance, vol 6 (2016) 31–35.
 - [14] Bizagi-Germany-GmbH, BAM Documentation, <https://help.bizagi.com/platform/en/index.html?bam.htm>, 2023. [Online; accessed 14-February-2025].

Surgery AI: Multimodal Process Mining and Mixed Reality for Real-time Surgical Conformance Checking and Guidance

Aleksandar Gavric, Dominik Bork and Henderik A. Proper

Business Informatics, TU Wien, Erzherzog-Johann-Platz 1, Vienna, 1040, Austria

Abstract

This paper discusses an end-to-end methodology for real-time surgical conformance checking that uses multimodal process mining, mixed reality (MR), and large language model (LLM) prompting. Our approach aims to support surgeons and medical teams by comparing *as-is* operational data—captured through a variety of sensors including MR-based gaze tracking—with a reference surgical process model encoded in Business Process Modeling Notation (BPMN). We illustrate how *shallow* and *deep* human-in-the-loop feedback mechanisms can be integrated with chain-of-thought prompting to provide relevant, context-aware, and iterative feedback during surgery. We further indicate which aspects of the surgery can be monitored (and hence queried) by our multimodal process mining engine. By enabling precise, actionable feedback during critical surgical procedures, our approach enhances the ability to identify deviations, ensure adherence to best practices, and reduce human error. Ultimately, this methodology empowers surgical teams to make data-driven adjustments, promotes better patient outcomes, and allows hospitals to monitor surgical conformance effectively, setting a new standard for process-driven healthcare assistance.

Keywords

Multimodal data analysis, Mixed reality, Surgery AI, Surgical guidance, Process mining, BPMN, LLM, Healthcare, assistant

1. Introduction

Modern surgical procedures are intricate and involve numerous steps, actors, instruments, and real-time decisions. Ensuring that each step in the *as-is* surgery conforms to a reference (or “desired”) model is crucial for patient safety, consistent outcomes, and compliance with institutional guidelines. Traditional methods of process oversight often rely on paper-based checklists or single-modality digital signals (e.g., time stamps of major milestones), which offer limited real-time insight.

Process mining [1] addresses this gap by extracting event logs from complex systems and reconstructing an *as-is* process model. Yet standard process mining may overlook the depth of real-time information available from modern medical devices, images, sensor data, and user interactions in an operating room [2]. The growing accessibility of mixed reality (MR) systems and advanced wearable sensors (like gaze trackers) opens the door to *multimodal*

✉ aleksandar.gavric@tuwien.ac.at (A. Gavric); dominik.bork@tuwien.ac.at (D. Bork);
henderik.proper@tuwien.ac.at (H. A. Proper)

ORCID 0009-0005-1243-7722 (A. Gavric); 0000-0001-8259-2297 (D. Bork); 0000-0002-7318-2496 (H. A. Proper)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

process mining [2, 3], where we capture a richer set of signals beyond textual or numeric logs (e.g., surgeon gaze, instrument position, physical environment changes, real-time vitals).

Meanwhile, Large Language Models (LLMs) allow us to harness *conversational* and *chain-of-thought* prompting to incorporate human expertise dynamically. Surgeons, nurses, and other staff can interact with the system at various depths: (1) *Shallow feedback*: Quick confirmations or corrections to immediate queries (e.g., “Is the incision completed?”), and (2) *Deep feedback*: More reflective input that leads to refining the underlying process model or augmenting the system’s domain knowledge [4].

Mixed reality interfaces can further project relevant information in the surgical environment, supporting *Spatial Conceptual Modeling* [5] to visualize conformance data in situ. This integration bridges the gap between human expertise and automated systems by enabling real-time contextual feedback and adaptive process modeling. For instance, visual overlays or auditory alerts can notify surgeons of deviations from standard procedures or highlight critical decision points, leveraging AI-based interpretation of multimodal data [6].

2. Related Work

The integration of artificial intelligence (AI) and mixed reality (MR) in surgical environments has emerged as a promising research area, driven by advancements in computer vision, language models, and multimodal process mining. This section reviews the most relevant contributions in this domain.

Recent efforts, such as Surgical-LLaVA [7], have demonstrated the potential of large language and vision models for understanding surgical scenarios, offering a foundation for enhanced decision support systems. Similarly, Yuan et al. [8] proposed a procedure-aware surgical video-language pretraining approach, utilizing hierarchical knowledge augmentation to improve the interpretability of surgical workflows. Digital twins, as described by Ding et al. [9], provide a unifying framework for surgical data science, leveraging geometric scene understanding to create comprehensive models of the operating room (OR). Complementing this, holistic OR domain modeling using semantic scene graphs has been explored by Özsoy et al. [10], enabling a detailed representation of surgical environments.

Further advancements in surgical scene graph knowledge have been achieved by Yuan et al. [11], who incorporated scene graphs into visual question answering (VQA) systems for surgical applications, thereby enhancing context-awareness in automated systems. The Ophnet benchmark by Hu et al. [12] provides a large-scale video dataset for ophthalmic surgical workflow understanding, facilitating the development of robust AI models in the domain.

Incorporating mixed reality into surgical planning and execution has also gained traction. Bracale et al. [13] highlighted the utility of MR in preoperative planning for colorectal surgery, showcasing its potential to improve surgical outcomes. From a conceptual perspective, Fill [5] introduced spatial conceptual modeling, which anchors knowledge in the physical world using augmented reality technologies, enabling innovative applications in medical and other domains.

Our prior contributions have laid the groundwork for advancing multimodal process mining and its applications. In Multimodal Process Mining [2], we introduced an approach to enrich traditional process mining with multimodal evidence, capturing data from diverse sources such

as sensors, images, and user interactions. Building on this, we explored how to enhance business process event logs with multimodal evidence in [3], demonstrating the potential for deeper insights. In [4], we addressed the challenge of tailoring multimodal data representations to stakeholder-specific terminology for improved interpretability. Finally, in [6], we extended the multimodal paradigm to conceptual modeling, showcasing how AI can leverage visual and auditory cues to interpret UML diagrams. These contributions collectively highlight the potential of multimodal approaches in augmenting traditional process and conceptual modeling practices.

By uniting algorithmic-symbolic rigor with LLM-driven sub-symbolic flexibility and human expertise, our approach transcends the constraints of rule-based process mining, enabling a more dynamic and contextually rich analysis of surgical workflows.

3. Methodology Overview

We formalize the multimodal process monitoring and adaptive feedback mechanism as an optimization problem over a hybrid state space \mathcal{S} consisting of structured process models, multimodal sensor inputs, and user feedback mechanisms.

State Representation Let the state at time t be represented as: $S_t = (M_t, X_t, U_t)$, where $M_t \in \mathcal{M}$ represents the current process model state (e.g., BPMN graph representation, stored in a Retrieval Augmented Graph [14]), $X_t \in \mathcal{X}$ denotes the vector of multimodal sensor observations (e.g., gaze tracking, instrument logs, voice commands), and $U_t \in \mathcal{U}$ captures the human feedback at time t , either shallow (e.g., confirmation) or deep (e.g., structural model changes).

Transition Function The state transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ maps the current state and action to a new state, $S_{t+1} = T(S_t, A_t)$ where A_t represents an action taken by the system or user, such as:

- A_t^S (System Actions): Process conformance checking, real-time alerting, adaptive workflow modification,
- A_t^H (Human Actions): Explicit feedback confirmation, model refinement, procedural adjustments.

Objective Function The system aims to minimize a cumulative deviation function J that quantifies non-conformance with the desired process model while maximizing the incorporation of human feedback. This ensures continuous process adaptation and human-in-the-loop refinement over time.

3.1. Application to a Specific Use Case: Surgery

We instantiate our proposed framework in the context of surgery, a domain characterized by strict procedural adherence and real-time decision-making.

Process Modeling and Sensor Integration The BPMN model for procedures includes pre-defined steps such as incision, trocar placement, laparoscope insertion, and organ manipulation. The system continuously maps real-world observations to this structured model through:

- Vision-based Instrument Detection (X_t^{inst}): Identifies tool usage and compares with expected sequences.
- Eye-tracking (X_t^{gaze}): Confirms if surgeons are focusing on critical areas at appropriate steps.
- Hand Gesture Recognition (X_t^{gest}): Detects compliance with required movements (e.g., correct suturing technique).
- Voice Commands (X_t^{voice}): Captures surgeon-nurse communications for validation.
- Real-time Imaging (X_t^{img}): Analyzes anatomical landmarks for correct procedure execution.

Illustrative Scenario Consider a scenario where a surgeon employs a novel technique requiring a secondary incision. The system detects a deviation (X_t^{img} and X_t^{inst} differ from the expected process).

Figure 1 provides a high-level schematic overview of our proposed framework adapted for the domain of surgery. Two major phases of human-in-the-loop involvement are depicted:

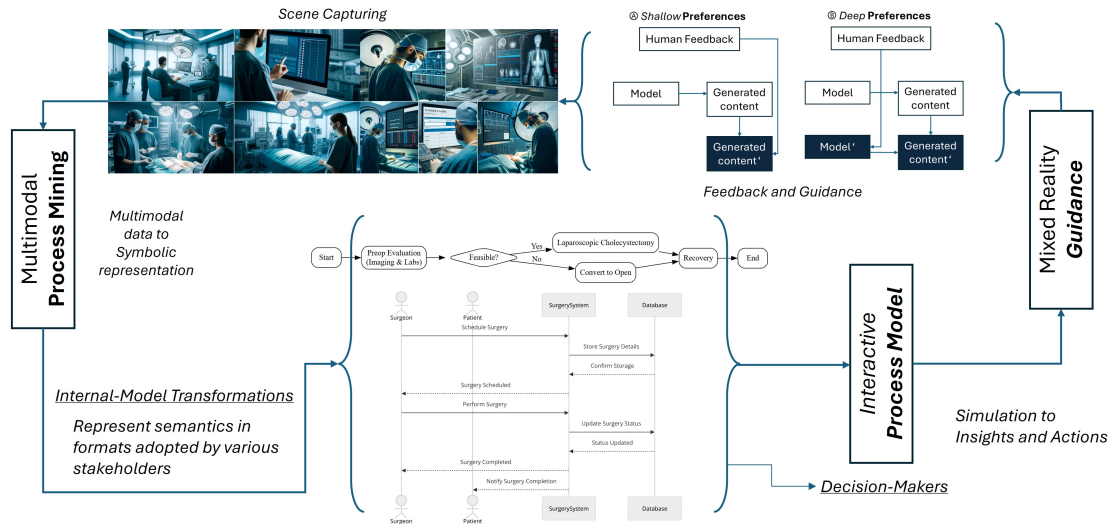


Figure 1: (Top) Illustration of human feedback via (A) Shallow and (B) Deep approaches, and a multimodal scene tracking. (Bottom) The solution's pipeline and illustrated (simplified) conceptual (process) model representation adjusted for conformance checking and guidance.

1. Shallow Feedback (A):

- The system continuously captures data from multiple sources (e.g., gaze tracking, instrument usage, sensor logs).
- It compares the *as-is* process to the *desired* BPMN model.

- When a potential discrepancy or question arises, the system prompts the user (surgeon, nurse, etc.) for feedback.
- The user provides confirmation, correction, or small clarifications. This feedback is used to adjust or annotate the *current* run-time process instance.

2. Deep Feedback (B):

- In-depth reflections by human experts are used to refine the *model* itself or the methods that interpret the captured data.
- For instance, if the current process model does not account for a new device or step introduced by the surgical team, deep feedback cycles can lead to an updated BPMN model or a reconfiguration of the data capture pipeline.
- Over time, repeated deep feedback loops result in an evolving knowledge base that is more robust and better tailored to each specific surgery or environment.

A critical component of our setup is the Mixed Reality (MR) environment, which serves multiple purposes:

- **Precision of Multimodal Recordings:** By using MR headsets, the system can track the surgeon’s gaze in relation to specific instruments or areas of the patient’s body. Likewise, position and orientation of surgical staff can be recorded.
- **Spatial Conceptual Modeling for Feedback:** We build on *Spatial Conceptual Modeling* [5], which allows us to overlay real-time process conformance data directly into the OR environment. For example, a soft highlight (visible in the MR headset) might appear over the next instrument to be used, or an alert icon might appear above a piece of equipment that must be sanitized.

3.2. Chain-of-Thought LLM Prompting

The proposed methodology employs a *conversation engine* powered by Large Language Models (e.g., GPT variants) that can: (1) **Parse sensor events** and interpret them in the context of the BPMN model, (2) **Generate feedback prompts** when conformance might be violated, (3) **Solicit clarifications and deeper insights** from the surgeon or nurse (for refining the model), and (4) **Provide intermediate “food-for-thought”** (chain-of-thought) to guide the surgical team or system designers on why certain steps are suggested or flagged.

A key component of our methodology is the construction of *well-curated LLM prompts* that merge domain knowledge (e.g., typical steps in a laparoscopic procedure) with real-time sensor data (e.g., the last tool recognized by a vision sensor was a cauterizing instrument). Below is a conceptual example of the layered prompts:

Example Prompt for Understanding Mixed Reality Inputs

System (LLM context): “The surgeon’s gaze has been fixated on the laparoscope for 5 seconds, and the nurse passed the laparoscope 10 seconds ago. The BPMN model indicates we are in the “Insert Laparoscope” task. Confirm if this step is complete. If uncertain, ask for feedback from the user.”

Example Prompt for Generating Feedback

System (LLM context after receiving user input): *“User indicated that they are testing a new technique requiring a secondary incision. The current BPMN model does not include this step. Rather than adding an optional sub-process, this should be modeled as an alternative process path. Insert an exclusive Gateway with the existing technique subprocess and the new technique subprocess as subsequent elements. Record new recommended tasks accordingly. Provide a revised BPMN snippet.”*

3.3. Aspects to Monitor for Multimodal Process Mining

Beyond the questions a surgeon might explicitly ask, the system continuously mines data to update the *as-is* process model. Table 1 outlines various aspects of surgery that are relevant for conformance checking, each corresponding to multimodal sensor inputs.

Table 1: Aspects to Monitor for Real-Time Surgical Conformance Checking

Aspect	Sensor/Data Source	Reason for Relevance to Conformance
Instrument Usage	Instrument detection via computer vision (camera) + staff input logs	To confirm correct usage sequence, detect missing/extra usage, alert if an instrument wasn't sterilized, etc.
Gaze Tracking	MR headset with eye-tracking	To assess if the surgeon is focused on the correct region/patient area. Non-conformance might arise if the surgeon fails to visually confirm a step (e.g., lack of inspection).
Hand Gestures	MR/IR sensors, glove-based trackers	To detect if certain steps (e.g., suturing technique) are performed in standard manner, or to confirm that a gesture-based command has been recognized.
Patient Vitals	Anesthesia machine logs, heart rate monitor, SpO2 sensor	To ensure anesthesia compliance steps, watch for anomalies that might require altering the process (e.g., emergency protocols).
Tool Count	Vision-based object detection, manual logs from nurses	To check if the correct number of instruments/sponges are present before closure (avoid retained surgical items).
Environment Sterility	UV sensor logs, staff compliance logs (handwashing, glove changes)	Conformance checking for infection control steps, verifying that each area is sanitized prior to the next step.

(continued on next page)

...(table continued)

Aspect	Sensor/Data Source	Reason for Relevance to Conformance
Surgeon/Nurse Position	MR device tracking (position/orientation)	To ensure correct posture or vantage point is taken for certain steps (e.g., for laparoscopic approach, a specific angle might be recommended).
Incision and Wound	Camera feed from laparoscope or overhead camera	To verify compliance with recommended incision size, location, and closure technique.
Anatomical Landmarks	Imaging data (e.g., real-time ultrasound, MRI overlays)	To confirm that the correct organ or region is identified before proceeding (e.g., right kidney instead of left).
Timeline / Timing	Digital clock + event logs	To confirm that each task is within an acceptable time window (e.g., prophylactic antibiotics repeated in time).
Communication Logs	Voice recognition or typed notes	To verify that critical verbal confirmations are done (e.g., "Time Out" procedure).
Clinical Documentation	EHR (Electronic Health Record) system	To confirm data entry is complete and consistent with the surgical plan (e.g., procedure codes, lab results).
Unexpected Events	Automatic anomaly detection (vitals, sudden camera movements)	To trigger re-routing of the BPMN process to an emergency sub-process if necessary (e.g., severe hemorrhage).
Surgeon/Staff Vitals	Smartwatches, wearable health trackers	To monitor the physical state of surgeons and staff (e.g., heart rate, stress levels, fatigue) and proactively suggest breaks or duty switches when signs of exhaustion or stress are detected, especially in surgeries involving multiple surgeons.

4. Proposed Evaluation and Future Work

We propose a multi-faceted evaluation framework, leveraging established surgical video datasets [15, 16, 17] to benchmark performance across several key metrics:

- **Annotation Accuracy:** Measure tool and event recognition accuracy against expert annotations.
- **Temporal Consistency:** Evaluate the alignment between detected events and ground truth timelines, ensuring timely alerts and correct sequencing.
- **Process Conformance:** Assess the system's ability to detect deviations from standard

protocols using conformance checking metrics, such as deviation frequency and critical event misclassifications.

- For evaluating the **robustness across modalities**, we will analyze performance consistency across different sensor inputs to ensure reliable multimodal integration.

By applying these metrics on diverse datasets from cataract [15], laparoscopic [16], and robotic surgery domains [17], we aim to demonstrate the system’s versatility and readiness for real-time surgical support.

Our roadmap for future work outlines key steps to ensure continuous improvement and user-centric development: (1) Extend evaluations to large, diverse datasets, including complex and rare surgical procedures, (2) implement rigorous testing on annotated datasets to validate real-time performance and scalability, and (3) engage with final users (surgeons, nurses) to gather feedback on system performance and usability.

5. Conclusion

By integrating *multimodal process mining* with *Mixed Reality* and *LLM-driven chain-of-thought prompting*, we propose a highly granular, real-time conformance checking methodology for surgical processes. User confirmations can augment immediate decisions in the operating room, while deeper reflection iteratively improves the process model over time.

As a result, surgeons can rely on the system to (1) provide step-by-step prompts and clarifications, (2) alert them when tasks are out of sequence or incomplete, (3) suggest new tasks when a procedure deviates from established protocols, and (4) support advanced analytics using chain-of-thought reasoning that ties sensor data to context-specific knowledge of surgical procedures.

Despite its promising capabilities, our approach has several limitations. Inaccuracies in sensor data (e.g., video feeds, gaze tracking) or inconsistent data quality may affect the system’s reliability. The methodology validated on selected surgical datasets, may require significant adaptation to perform effectively across diverse surgical procedures and environments. Achieving true real-time performance can be challenging due to the computational complexity of multimodal data fusion and chain-of-thought processing.

Addressing these threats and limitations through continued testing, iterative user feedback, and technological refinements will be essential for future deployments in dynamic surgical environments.

References

- [1] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Springer, Berlin, Heidelberg, 2011. URL: <https://doi.org/10.1007/978-3-642-19345-3>. doi:10.1007/978-3-642-19345-3.
- [2] A. Gavric, D. Bork, H. Proper, Multimodal process mining, in: *26th International Conference on Business Informatics (CBI)*, 2024.

- [3] A. Gavric, D. Bork, H. Proper, Enriching business process event logs with multimodal evidence, in: The 17th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM), 2024.
- [4] A. Gavric, D. Bork, H. Proper, Stakeholder-specific jargon-based representation of multimodal data within business process, in: Companion Proceedings of the 17th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM Forum 2024), 2024.
- [5] H.-G. Fill, Spatial Conceptual Modeling: Anchoring Knowledge in the Real World, Springer Nature Switzerland, Cham, 2024, pp. 35–50. URL: https://doi.org/10.1007/978-3-031-56862-6_3. doi:10.1007/978-3-031-56862-6_3.
- [6] A. Gavric, D. Bork, H. Proper, How does uml look and sound? using ai to interpret uml diagrams through multimodal evidence, in: 43rd International Conference on Conceptual Modeling (ER), 2024.
- [7] J. Jin, C. W. Jeong, Surgical-llava: Toward surgical scenario understanding via large language and vision models, 2024. URL: <https://arxiv.org/abs/2410.09750>. arXiv: 2410.09750.
- [8] K. Yuan, V. Srivastav, N. Navab, N. Padoy, Procedure-aware surgical video-language pretraining with hierarchical knowledge augmentation, arXiv preprint arXiv:2410.00263 (2024).
- [9] H. Ding, L. Seenivasan, B. D. Killeen, S. M. Cho, M. Unberath, Digital twins as a unifying framework for surgical data science: the enabling role of geometric scene understanding, *Artificial Intelligence Surgery* 4 (2024) 109–138.
- [10] E. Özsoy, T. Czempiel, E. P. Örnek, U. Eck, F. Tombari, N. Navab, Holistic or domain modeling: a semantic scene graph approach, *International Journal of Computer Assisted Radiology and Surgery* 19 (2024) 791–799.
- [11] K. Yuan, M. Kattel, J. L. Lavanchy, N. Navab, V. Srivastav, N. Padoy, Advancing surgical vqa with scene graph knowledge, *International Journal of Computer Assisted Radiology and Surgery* (2024) 1–9.
- [12] M. Hu, P. Xia, L. Wang, S. Yan, F. Tang, Z. Xu, Y. Luo, K. Song, J. Leitner, X. Cheng, et al., Ophnet: A large-scale video benchmark for ophthalmic surgical workflow understanding, in: *European Conference on Computer Vision*, Springer, 2025, pp. 481–500.
- [13] U. Bracale, B. Iacone, A. Tedesco, A. Gargiulo, M. M. Di Nuzzo, D. Sannino, S. Tramontano, F. Corcione, The use of mixed reality in the preoperative planning of colorectal surgery: Preliminary experience with a narrative review, *Cirugía Española (English Edition)* (2024).
- [14] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. arXiv:2005.11401.
- [15] H. Al Hajj, M. Lamard, P.-H. Conze, S. Roychowdhury, X. Hu, G. Maršalkaitė, O. Zisimopoulos, M. A. Dedmari, F. Zhao, J. Prellberg, M. Sahu, A. Galdran, T. Araújo, D. M. Vo, C. Panda, N. Dahiya, S. Kondo, Z. Bian, A. Vahdat, J. Bialopetravičius, E. Flouty, C. Qiu, S. Dill, A. Mukhopadhyay, P. Costa, G. Aresta, S. Ramamurthy, S.-W. Lee, A. Campilho, S. Zachow, S. Xia, S. Conjeti, D. Stoyanov, J. Armaitis, P.-A. Heng, W. G. Macready, B. Cochener, G. Quelled, Cataracts: Challenge on automatic tool annotation for cataract surgery, *Medical Image Analysis* 52 (2019) 24–41. doi:<https://doi.org/10.1016/j.media.2018.11.008>.
- [16] A. P. Twinanda, S. Shehata, D. Mutter, J. Marescaux, M. de Mathelin, N. Padoy, Endonet: A deep architecture for recognition tasks on laparoscopic videos, 2016. arXiv:1602.03012.

- [17] M. Allan, S. Kondo, S. Bodenstedt, S. Leger, R. Kadkhodamohammadi, I. Luengo, F. Fuentes, E. Flouty, A. Mohammed, M. Pedersen, A. Kori, V. Alex, G. Krishnamurthi, D. Rauber, R. Mendel, C. Palm, S. Bano, G. Saibro, C.-S. Shih, H.-A. Chiang, J. Zhuang, J. Yang, V. Iglovikov, A. Dobrenkii, M. Reddiboina, A. Reddy, X. Liu, C. Gao, M. Unberath, M. Kim, C. Kim, C. Kim, H. Kim, G. Lee, I. Ullah, M. Luna, S. H. Park, M. Azizian, D. Stoyanov, L. Maier-Hein, S. Speidel, 2018 robotic scene segmentation challenge, 2020. [arXiv:2001.11190](https://arxiv.org/abs/2001.11190).

Author Index

Arnold, Lisa, 39, 48

Breitmayer, Marius, 48

Böhm, Sebastian, 25, 32

Frisch, Anton, 16

Ghani, Mustafa, 12

Hehnle, Philipp, 1

Heinze, Thomas, 8

Koller, Jakob, 32

Kudravcev, Roman, 25

Lichtenthäler, Robin, 16

Reichert, Manfred, 1, 48