Sebastian Böhm
Daniel Lübke

ZEUS 2023

15th ZEUS Workshop, ZEUS 2023,
Hannover, Germany, 16–17 February 2023
Proceedings

Volume Editors

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

Daniel Lübke
Digital Solution Architecture

Preface

In February 2023, we had the pleasure to organize the 15^{th} edition of the ZEUS Workshop planned in Hannover, Germany. This time, the workshop was held on-site again, giving us the chance to meet and discuss up-to-date research in person. At this point, we would like to thanks a lot all reviewers for their work and the ongoing support.

This workshop series offers young researchers an opportunity to present and discuss early ideas and work in progress as well as to establish contacts among young researchers. For this year's edition, we selected all nine submissions for presentation at the workshop. Each submission went through a thorough peer-review process and was assessed by at least three members of the program committee with regard to its relevance and scientific quality. The accepted contributions cover the areas of Business Process Management, Cloud Computing, Microservices, Software Design, and the Internet of Things.

The workshop was generously sponsored by Camunda Services GmbH.

Hannover, February 2023

Sebastian Böhm
Daniel Lübke

Organization

Steering Committee

Nico Herzberg	Campeleon GmbH
Oliver Kopp	Daimler AG
Stefan Kolb	JabRef Research
Stephan Haarmann	Hasso Plattner Institute, University of Potsdam
Johannes Manner	University of Bamberg

Local Organizer

Daniel Lübke	Digital Solution Architecture

Web Chair

Robin Lichtenthäler	University of Bamberg
Sebastian Böhm	University of Bamberg

Program Committee Chairs

Sebastian Böhm	University of Bamberg

Program Committee

Saimir Bala	Vienna University of Economics and Business
Marius Breitmayer	University of Ulm
Achim D. Bruckner	University of Exeter
Jonas Cremerius	Hasso Plattner Institute, University of Potsdam
Stephan Fahrenkrog-Peterson	Humbodt-Universität Berlin
Manuel Fritz	University of Stuttgart
Georg Grossmann	University of South Australia
Lukas Harzenetter	University of Stuttgart
Thomas Heinze	German Aerospace Center
Pascal Hirmer	University of Stuttgart
Christoph Hochreiner	Compass Verlag
André van Hoorn	University of Hamburg
Martin Kabierski	Humboldt-Universität zu Berlin
Simone König	Mercedes-Benz AG, TU Munich
Jan Ladleif	Hasso Plattner Institute, University of Potsdam
Jörg Lenhard	SAP SE
Robin Lichtenthäler	University of Bamberg
Daniel Lübke	Digital Solution Architecture
Matteo Nardelli	University of Rome Tor Vergata
Adrian Rebmann	University of Mannheim
Fabiana Rossi	University of Rome Tor Vergata
Jan Sürmeli	FZI Forschungszentrum Informatik, Karlsruhe
Maximilian Völker	Hasso Plattner Institut
Tom Lichtenstein	Hasso Plattner Institut
Stefan Winzinger	University of Bamberg

Sponsoring Institutions

Camunda Services GmbH

Contents

Model Reader Preferences for Semantically Duplicate Elements in BPMN

Daniel Lübke[1,2,*], Volker Stiehl[3]

[1]Digital Solution Architecture GmbH, Hannover, Germany
[2]Leibniz Universität Hannover, FG Software Engineering, Hannover, Germany
[3]TH Ingolstadt, Ingolstadt, Germany

Abstract

BPMN, which is the underlying modeling notation of many BPM endeavours and business information system development projects, is a rich modeling language, which also offers redundant constructs, i.e., different syntax can express the same semantics. We want to investigate which syntactical constructs are preferred by model readers if different ways to model message exchanges are offered by BPMN. In an empirical study we asked 77 participants which BPMN model they prefer for expressing eight situations. We found that send tasks and intermediate message catch events are significantly preferred. Also, event-based gateways are preferred over boundary events for many variants of the Deferred Choice pattern.

Keywords

BPMN, Empirical Study, Gateway, Boundary Event, Message, Subjective Preference, Event-based Gateway

1. Motivation

BPMN is THE standard for modeling business processes. Nowadays, business-critical applications based on BPMN and modern architectures [1, 2] are developed to digitize important business processes. Consequently, BPMN is used to communicate between a variety of stakeholders, e.g., developers and business analysts, and thus understandability is very important. While BPMN offers a wide set of modeling options for expressing many process details, it contains redundant constructs. For example, modeling message arrival time-outs can be modeled in different ways as explained in this paper. Allowing ambiguity how to model a certain situation allows for confusion and misunderstandings. Consequently, clarifying the usage of redundant syntax could standardize the current use of BPMN, streamline future versions of BPMN and thus make the notation easier to learn and understand. This paper presents a first step into this direction by investigating the subjective preferences of a) modeling message-based communication and b) representations of the deferred choice workflow pattern [3], when messages are involved. This paper is structured as follows. Within the next Section we present related

ZEUS'2023: S15th Central European Workshop on Services and their Composition, February 16–17, 2023, Hannover, Germany

*Corresponding author.
daniel.luebke@digital-solution-architecture.com (D. Lübke); volker.stehl@thi.de (V. Stiehl)
https://www.digital-solution-architecture.com (D. Lübke)
0000-0002-1557-8804 (D. Lübke)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

work. In Section 3 the design of our empirical study is presented. The results are presented in Section 4 and an interpretation of those are given in Section 5. Finally, we conclude and give an outlook.

2. Related Work

Quality of business process models is multi-faceted. Lindland et al. [4] specified a framework that can be used to categorize different quality aspects of models, in which they distinguish between syntactic, semantic, and pragmatic qualities. This paper is concerned with subjective preference of certain model constructs. Because "[i]n general, researchers associate aesthetics with readability, and readability with understanding" [5] subjective preference is a part of *understandability* and thus a pragmatic quality. Or as Lindland et al. put it: Understandability is the main concern of pragmatic model quality, which "affects how to choose from among the many ways to express a single meaning" [4]. Comprehension of BPMN models is a vast research area: For example, there are studies concerning the influence of layout on understandability. Figl provides a good overview [6]. Scholz & Lübke [7] investigated subjective layout preferences and used the same research design as we do: By using a quiz-like study, in which participants choose one of the presented options, they have analyzed subjective preferences of different choices for BPMN layouts. Moody [8] has critiqued BPMN in general for failing to adhere to his "Physics of Notations" [9] – especially that BPMN has considerable semantic redundancy, e.g., the *Exclusive OR Gateway* has two visual representations. Genon et al. [10] found the same. The eCH-0158 modeling guidelines for BPMN [11] recognize the redundancy between *send/receive tasks* and *message catching/throwing events*. They standardize on *send tasks* and *message catch events*.

3. Study Design

3.1. Goals, Hypothesis & Variables

By following the Goal-Question-Metric (GQM) approach [12] we are defining our goal as

Understand the *Subjective Preference* with regard to *Semantically Equivalent Elements in BPMN 2.0* from the viewpoint of a *Model Reader*.

This goal is refined into (research) questions. While BPMN has many redundancies, we concentrate on the ones below. We want to answer, which construct for each of the following pairs of semantically equivalent BPMN constructs are preferred:

RQ1 Send Task vs. Intermediate Message Throw Event: BPMN offers two elements for sending messages: The *send task* and the *intermediate message throw event* both send a message.

RQ2 Receive Task vs. Intermediate Message Catch Event: Similarily to sending a message, BPMN also offers a *receive task* and an *intermediate message catch event for receiving*

a message.

RQ3 Send Task vs. End Message Throw Event: For modeling the sending of a message at the end of a process execution, a *send task* and a *none end event* can be used. Alternatively, an *message throw end event* can be used.

RQ4 Deferred Choice between two messages (diff. prob.): A Deferred Choice [3] between two incoming messages can be modelled via an *event-based gateway* or a *receive task* with an *interrupting message boundary event*. Because one participant in [7] indicated that he/she would model splits and joins differently depending on the probability of the branch taken, we differentiate between the probability of events. This question is concerned with messages that have different probabilities, i.e., the top *event* after the *event-based gateway* and the message caught by the *receive task* are more likely to occur than the bottom *event*, which is more exceptional, after the *event-based gateway* and the message caught by the *boundary event*.

RQ5 Deferred Choice between two messages (same prob.): This question is similar to RQ4. However, the incoming messages have the same probability, i.e., both events following the *event-based gateway* and both messages occur equally often.

RQ6 Deferred Choice between message and timer (diff. prob.): This question is similar to RQ4 but this time the Deferred Choice is not between two messages but instead resembles a deadline situation with a *message event* and a *timer event*. It is more probable to receive the message than to time-out. This pattern is presented as an *event-based gateway* with two following *events* or with a *receive task* with an *interrupting timer boundary event*.

RQ7 Deferred Choice between message and timer (same prob.): This question is similar to RQ6. However, the incoming message and the time-out have the same probability.

RQ8 Deferred Choice between two messages and a timer: The last question is concerned with a Deferred Choice between two messages and a timer, i.e., a scenario in which one of two messages must be received within a certain time. This can – again – be modeled as an *event-based gateway* followed by two *message events* and one *timer event*, or by a *receive task* with two *boundary events*.

3.1.1. Measurements & Hypothesis

We measure the subjective preferences of study participants as the only metric for all research questions. For all research questions the null hypothesis H_0 is that there is no preference for one of the two alternatives. Accordingly, H_1 is that one of the two alternatives is preferred.

3.2. Objects

The study setup is similar to a previous study by Scholz & Lübke [7]: Participants take part in an online survey in which two diagrams modeling the same process are shown which only

Table 1
Description of the Participants Groups of our Study

Group	Experience	Description	Count
LUH1	Students	MSc./CS, Software Architecture Lecture	20
LUH2	Students	BSc./CS, Software Engineering Seminar	3
LUH3	Students	MSc./CS, Software Methodologies Lecture	4
THI1	Students	BSc./IS, 4th semester	11
THI2	Students	BSc./IS, 6th semester	6
THI3	Students	MSc./IE, 2nd semester	9
THI4	Students	BSc./IE, 6th semester	11
Prof.	Professionals	recruited from different organizations	13
Total			**77**

differ in one point. In this study different but semantically equivalent BPMN diagrams were used as shown in Appendix A. Both options were shown side by side and participants had to choose the preferred one by clicking it. Descriptive text was shown to convey the probablity of some branches. Since branching probabilities cannot be modeled in BPMN directly, it was necessary to convey this information textually.

3.3. Participants

Participants were a) recruited from lectures of the authors and b) professionals were asked to participate. We tracked the group to which a participant belongs to by using different invitation links. Participation was voluntary and no incentives were given. The number of participants per group and a more detailed description is shown in Table 1. All in all, we had 87 participants in total. After removing those, who did not complete the quiz or changed their answers in between, 77 participants remained.

3.4. Validity Procedure

As a first step we performed a power test: For a two-sided hypothesis test with $\alpha = p = 0.05$ and confidence $\beta = 0.95$ for a medium effect of $h = 0.5$ yields that we required at least 52 participants. As described above we recruited more participants than required. For eliminating extraneous variables we took following measures: We randomized the order in which questions (i.e., diagram pairs) were shown. Thereby, we try to eliminate learning and fatigue effects. We also randomized the order in which diagrams are shown.

4. Analysis

The statistical evaluation of the gathered data is shown in Table 2. The statistical significance indicated by the p-values is marked by asterisks (*: $p \leq 0.05$, **: $p \leq 0.01$, ***: $p \leq 0.001$). Similarily, the effect is denoted by pluses (+: $h \geq 0.2$, ++: $h \geq 0.5$, +++: $h \geq 0.8$).

Table 2
Results and Hypothesis Test Results for all Questions

	Question	#A	#B	p	*	h	+
Q1	Send Task vs. Message Throw Event	53	24	0.0013	**	0.39	+
Q2	Receive Task vs. Message Catch Event	29	48	0.0395	*	0.25	+
Q3	Send Task vs. Message End Event	39	38	1.0000		0.01	
Q4	Deferred Choice, 2 messages, diff. prob. Gateway vs. Boundary Event	53	24	0.0013	**	0.39	+
Q5	Deferred Choice, 2 messages, same prob. Gateway vs. Boundary Event	69	8	0.0000	***	0.91	+++
Q6	Deferred Choice, message+timer, diff. prob. Gateway vs. Boundary Event	36	41	0.6488		0.06	
Q7	Deferred Choice, message+timer, same prob. Gateway vs. Boundary Event	43	34	0.3620		0.12	
Q8	Deferred Choice, 2 messages+timer Gateway vs. Boundary Events	58	19	0.0000	***	0.53	++

5. Interpretation

5.1. Evaluation of Results & Implications

The *send task* is significantly preferred over a *message throw event* (RQ1). It seems that participants see the sending of a message more as a task, i.e., an active action, and therefore prefer the task instead of an event.

In contrast to RQ1, participants significantly prefer a *message catch event* for waiting on a message receive (RQ2). Interestingly, it is inconsistent to use different syntax for sending and receiving messages. This can mean that perhaps participants differentiate between active and passive/waiting elements.

There is no significant difference for sending a message at the process end (RQ3). In contrast to a significant preference for a *send task* during the process, there is no clear preference for a *send task* with an *end event* or a *message end event*. It seems that the additional penalty of a second symbol and its associated space requirements is not worth to keep up the semantic difference experienced in RQ1.

When modeling a Deferred Choice between two messages which arrive with different probabilities, participants prefer the use of an *event-based gateway* (RQ4). It may be that the visuals of two white envelopes – one in the *receive task* and one in the *boundary event* – is not attractive. Participants have an even stronger preference for the *gateway* if the probability of the messages are the same (RQ5).

When modeling a time-out, i.e., a Deferred Choice between a message and a timer, neither the *gateway* nor the *boundary event* is preferred – regardless of whether the timer is as likely to occur (RQ6) or is only triggered as an exception (RQ7). This contrasts with the results from RQ4/5, which are structurally the same but use a different second event. While more participants liked the *gateway* for same probabilities of events and more participants liked the *boundary event* for exceptional cases, these differences were not significant. More research has to further

clarify whether there is a difference with a small effect or not.

If the Deferred Choice is between two messages and a timer event (RQ8) there is a strong, significant preference to the *event-based gateway*. However, we cannot attribute to why this is: While in our study planning we wanted to examine the effect of a larger number of boundary events, another possible explanation is that a send task with a message boundary event is disliked as RQ4 and RQ5 have shown.

5.2. Limitations of Study

Because we only measured subjective preferences no quantative data on model comprehension could be measured. This study still gives insights into model perception, especially with different variants of the Deferred Choice pattern. Like all studies which include students, the question of generalizability arises. However, we have seen that no differences between our groups exist – this also means that the group of professionals does not behave significantly different from the students. While we had a considerable amount of participants, some research questions gave non-significant results with a small effect size in the range of $0.1 \leq h \leq 0.2$. To have adequate power in the statistical tests, more participants (approx. 350) are required.

6. Implications for Practitioners

Following from these results practitioners should amend existing modeling guidelines by the following rules: 1) Use *Send Tasks* for sending messages during process execution, 2) use *Message Catch Events* for receiving messages during process execution, and 3) use *Event-based Gateways* when implementing the Deferred Choice pattern when receiving multiple messages. Modelers should keep in mind that this is the first study to examine these constructs. Hopefully, future studies will strengthen or refute these results and thus these proposed modeling guidelines.

7. Conclusions & Outlook

Within this paper we presented our empirical study with students from two universities and professionals on the subjective preference of syntactically redundant, message-related constructs in BPMN. We found a strong subjective preference for *send tasks* over *message throw events* within the process-flow and for *message catch events* over *receive tasks*. We also found that Deferred Choices in *event-based gateways* are preferred over *boundary events* in the case of two message events or three events. We could find no significant preference for Deferred Choices with a message and a timer ("time-outs") or for the sending of a message on process completion. While the results are interesting in themselves, this study lays the foundation for further empirical inqueries: Follow up studies, especially experiments, can investigate and compare understandability of redundant BPMN message-related constructs. This way, especially eye-tracking experiments, can be used to gather quantative data to evaluate whether the subjective preferences match the differences in objective understandability in the future, and further developing modeling guidelines for BPMN.

Acknowledgments

We'd like to thank all participants who took part in our study. Additionally, we like to thank Kurt Schneider, Dieter Kähny, and Barbara Ulrich for distributing the quiz within their classes and organizations.

References

[1] V. Stiehl, Implementing the Basic Architecture of Process-Driven Applications, Springer, 2014. URL: http://link.springer.com/chapter/10.1007/978-3-319-07218-0_4.

[2] B. Rücker, 3 common pitfalls in microservice integration and how to avoid them, WWW: https://berndruecker.io/3-pitfalls-in-microservice-integration/, last access: 2021-02-18, 2018.

[3] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow patterns, Distributed and parallel databases 14 (2003) 5–51.

[4] O. I. Lindland, G. Sindre, A. Solvberg, Understanding quality in conceptual modeling, IEEE Software 11 (1994) 42–49. doi:10.1109/52.268955.

[5] C. Bennett, J. Ryall, L. Spalteholz, A. Gooch, The aesthetics of graph visualization, Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging (2007) 57–64. doi:10.2312/COMPAESTH/COMPAESTH07/057-064.

[6] K. Figl, J. Recker, Exploring cognitive style and task-specific preferences for process representations, Requirements Engineering 21 (2016) 63–85. URL: http://dx.doi.org/10.1007/s00766-014-0210-2. doi:10.1007/s00766-014-0210-2.

[7] T. Scholz, D. Lübke, Improving automatic bpmn layouting by experimentally evaluating user preferences, in: Á. Rocha, H. Adeli, L. P. Reis, S. Costanzo (Eds.), New Knowledge in Information Systems and Technologies, Springer International Publishing, Cham, 2019, pp. 748–757.

[8] D. L. Moody, Why a Diagram is Only Sometimes Worth a Thousand Words: An Analysis of the BPMN 2.0 Visual Notation, 2011.

[9] D. L. Moody, The physics of notations: toward a scientific basis for constructing visual notations in software engineering, Software Engineering, IEEE Transactions on 35 (2009) 756–779.

[10] N. Genon, P. Heymans, D. Amyot, Analysing the cognitive effectiveness of the bpmn 2.0 visual notation, in: B. Malloy, S. Staab, M. van den Brand (Eds.), Software Language Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 377–396.

[11] A. Birchler, E. Bosshart, M. Märki, P. Opitz, J. Pauli, B. Rigert, Y. Sandoz, M. Schaffroth, N. Spöcker, C. Tanner, K. Walser, T. Widmer, eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung, WWW: https://www.ech.ch/dokument/fb5725cb-813f-47dc-8283-c04f9311a5b8, 2014. URL: https://www.ech.ch/dokument/fb5725cb-813f-47dc-8283-c04f9311a5b8.

[12] V. R. Basili, Applying the goal/question/metric paradigm in the experience factory, Software Quality Assurance and Measurement: A Worldwide Perspective (1993) 21–44.

A. Diagrams

Table 3
Diagrams used in our Study

RQ1	Send Task	Message Throw Event
RQ2	Receive Task	Message Catch Event
RQ3	Send Task	End Message Event
RQ4	Gateway (diff. prob.)	Boundary Event (Msg.)
RQ5	Gateway (same prob.)	Boundary Event (Msg.)
RQ6	Gateway (diff. prob.)	Boundary Event (Timer)
RQ7	Gateway (same prob.)	Boundary Event (Timer)
RQ8	Gateway	Boundary Event (Timer+Event)

Enhancing BPMN 2.0 with IoT Modeling Aspects: How Much Language is Enough?

Yusuf Kirikkayis[1], Florian Gallik[1] and Manfred Reichert[1]

[1]*Institute of Databases and Information Systems, Ulm University, 89081, Germany*

Abstract

Extending Business Process Management (BPM) with Internet of Things (IoT) capabilities enables real-world aware process automation, business rule execution, and process monitoring. As a prerequisite for exploiting the benefits of this real-world awareness, IoT behavior needs to be captured in business process models. The de facto standard for business process modeling, BPMN 2.0 seems to be appropriate for covering IoT aspects as well. However, modeling IoT-aware processes might be hindered by the ambiguous use of the BPMN 2.0 modeling elements. Still it has to be evaluated how process model readers actually perceive IoT aspects captured in BPMN 2.0 process models. This paper discusses the challenges of modeling IoT-aware business processes with BPMN and derives research questions to be investigated in future work.

Keywords

IoT-aware business process, BPM, IoT, BPMN 2.0

1. Introduction

The IoT represents a network of interconnected physical devices, i.e., sensors and actuators, that allow capturing, exchanging, and collecting data to respond to physical events. Thus, the dynamic context of the physical world can be captured and transformed to a digital shadow. The IoT, therefore, is a fundamental technology in areas likes smart manufacturing, smart logistics, or smart healthcare. In these areas IoT-aware business process support can create a competitive edge by exploiting the data produced by IoT devices [1][2]. We refer to processes that utilize IoT devices and map IoT behavior to process activities and events as IoT-aware.

The incorporation of IoT capabilities into IoT-aware business processes offers promising perspectives for bridging the gap between digital processes and the physical world [3]. While IoT enables collecting and exchanging data about the physical world, Business Process Management (BPM) enables modeling, implementing, executing, monitoring, and analyzing business processes [3]. In the context of BPM, moreover, IoT devices can be used to automate different types of tasks, to enhance process and task monitoring, and to support real-world aware decision-making [4]. An essential challenge of modeling IoT-aware business processes is to properly capture IoT-related aspects in the process models [5, 6]. Amongst others, modeling

ZEUS2023: Central European Workshop on Services and their Composition, February 16-17, 2023 in Hannover, Germany
yusuf.kirikkayis@uni-ulm.de (Y. Kirikkayis); florian-1.gallik@uni-ulm.de (F. Gallik); manfred.reichert@uni-ulm.de (M. Reichert)
0000-0001-6536-0785 (Y. Kirikkayis); 0000-0003-0861-275X (F. Gallik); 0000-0003-2536-4153 (M. Reichert)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

IoT aspects shall foster the understanding of how the process works and facilitate the discovery of potential problems (e.g., deadlocks).

As many of the BPMN 2.0 modeling elements can also be found in IoT-aware business processes, researchers have argued that BPMN 2.0 is capable of modeling IoT-aware business processes [4, 7, 8, 9, 10]. As example consider Figure 1, which depicts an IoT-aware light control process expressed in terms of the BPMN 2.0.

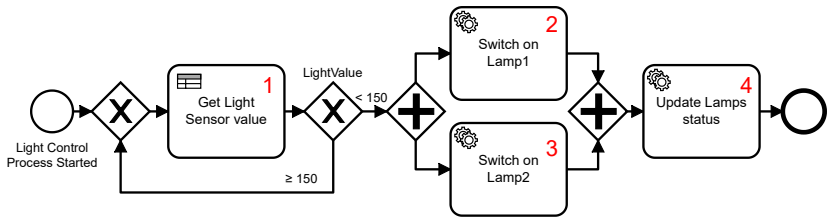


Figure 1: Example of an IoT-aware process model expressed in terms of BPMN 2.0.

Other works have explicitly indicated the need for modeling IoT participation in business processes [6, 11, 12, 13]. In particular, they recommend to visually distinguish between common and IoT-related modeling elements by extending the BPMN 2.0 meta-model with IoT-specific modeling elements. Consequently, the involvement of IoT devices in process enactment becomes apparent, fostering the comprehensibility of IoT-related process models and, thus, their maintenance. Corresponding works further argue that due to the extension of BPMN 2.0 with IoT-specific modeling elements, no ambiguities occur when reading process models. In Figure 1, Tasks 2&3 are IoT-related, whereas this does not apply to Task 4.

Modeling IoT-aware business processes with BPMN 2.0 has been extensively studied in literature [4, 7, 8, 9, 10, 14]. However, the evaluation of IoT-aware process models from a user perspective is still missing. In particular, we are interested (1) whether IoT-related processes modeled in terms of the standard BPMN 2.0 notation are as comprehensible as (2) IoT-aware processes modeled with BPMN 2.0 and explicit IoT-specific elements, with the latter constituting an extension of BPMN 2.0. A particular challenge is to identify those factors that foster the understanding of IoT involvement in BPMN 2.0 process models from a user perspective. To determine whether IoT aspects in BPMN 2.0 process models are properly recognized by users, human cognition and mental effort needs to be considered when reading corresponding process models. Related works neither cover the perspective of IoT-aware process models nor cognitive aspects of understanding corresponding models. In particular, decisions on how IoT aspects shall be captured in BPMN process models have been primarily based on technical issues. This paper derives research questions to investigate how IoT aspects shall be captured in BPMN process models taking the user perspective into account as well.

2. Research Questions

Although BPMN 2.0 offers various elements (e.g., *Activities*, *Events*, *Pools*, and *Lanes*) that allow modeling IoT-aware business processes, different aspects need to be considered. In particular,

additional research is required to investigate how IoT aspects shall be visually covered in BPMN 2.0 models, while fostering model comprehensibility (i.e. whether to opt for variant (1) or (2)) [15, 16]. In detail, the following research questions (RQs) need to be considered to understand whether a BPMN 2.0 extension for IoT-aware processes becomes necessary. Note that these research questions were derived from systematic literature reviews [17, 18] as well as a comparison of the two modeling approaches, i.e., modeling IoT-aware processes with standard BPMN (1) vs. modeling them based on an extension of BPMN with IoT-specific elements (2).

RQ 1	Is there any IoT-specific behavior that cannot be modeled in terms of BPMN 2.0?
RQ 2	Can IoT-related modeling elements be identified in BPMN 2.0 from a user perspective?
RQ 3	Are there patterns in modeling IoT-driven business processes?
RQ 4	Does the use of different BPMN modeling elements influence the cognitive load during the comprehension of IoT-aware processes?

RQ1 intends to investigate whether there exists any IoT-specific behavior which is relevant for process execution, but cannot be properly represented in terms of BPMN 2.0. To answer *RQ 1*, IoT-aware processes from different domains need to be analyzed and modeled with standard BPMN 2.0. In particular, this might unravel IoT behavior that cannot be directly modeled in BPMN 2.0. In contrast, *RQ 2* aim to identify whether IoT aspects captured in BPMN 2.0 process models can be visually or textually recognized from a user perspective when using standard modeling elements (e.g., pools, lanes, activity types, and event types). *RQ 3* aim to determine whether there are patterns of IoT-driven business processes when modeling them in terms of BPMN 2.0. In turn, *RQ4* investigates the cognitive load of users when reading and comprehending IoT-aware processes in BPMN 2.0. For this purpose, for example, a NASA-TLX questionnaire may be used. Answering *RQs 2 - 4* shall allow us to understand how IoT-aware business processes modeled in terms of standard BPMN 2.0 affect human cognition and how far the respective process models are perceived as IoT-aware from a user perspective. All four research questions need to be answered to assess whether BPMN 2.0 is suitable for modeling IoT-aware business processes or extension make sense to foster real-world aware processes.

3. Conclusions

This paper introduced four research questions that need to be addressed when modeling IoT-aware business processes with BPMN 2.0. In literature, two approaches are proposed for modeling IoT-aware processes: (1) using standard BPMN 2.0 as (2) extending the BPMN 2.0 standard with IoT-specific modeling elements. However, existing works have neglected the user perspective when deciding which of these two variants shall be used. Consequently the pros and cons on how to model IoT aspects in BPMN-based processes have primarily been considered form a technical perspective taken by IoT experts. User studies are needed that address the presented research questions. In corresponding studies, different aspects such as the recognition of IoT aspects in BPMN-based process models, the cognitive load of users when reading IoT-aware processes in BPMN 2.0, and different modeling patterns for IoT-aware processes in BPMN from user perspective can be identified. Various techniques may be used for this purpose, such as the survey of study participants and conducting a within-subject study.

References

[1] F. Martins, D. Domingos, Modelling iot behaviour within bpmn business processes, Procedia Computer Science (2017) 1014–1022. Int Conf on ENTERprise Information Systems.

[2] Y. Kirikkayis, F. Gallik, M. Reichert, Iotdm4bpmn: An iot-enhanced decision making framework for bpmn 2.0, in: Int Conf on Service Science, 2022, pp. 88–95.

[3] Janiesch, et al., The internet of things meets business process management: a manifesto, Systems, Man, and Cybernetics Magazine (2020) 34–44.

[4] R. Seiger, L. Malburg, B. Weber, R. Bergmann, Integrating process management and event processing in smart factories: A systems architecture and use cases, Journal of Manufacturing Systems (2022) 575–592.

[5] Y. Kirikkayis, F. Gallik, M. Reichert, Modeling, executing and monitoring iot-driven business rules with bpmn and dmn: Current support and challenges, in: Enterprise Design, Operations, and Computing, Springer Int Publishing, 2022, pp. 111–127.

[6] Y. Kirikkayis, F. Gallik, M. Reichert, Towards a comprehensive bpmn extension for modeling iot-aware processes in business process models, in: Research Challenges in Information Science, Springer Int Publishing, 2022, pp. 711–718.

[7] F. Hasić, E. S. Asensio, Executing iot processes in bpmn 2.0: Current support and remaining challenges, in: Research Challenges in Information Science, 2019, pp. 1–6.

[8] A. Caracas, From business process models to pervasive applications: Synchronization and optimization, in: Int Conf on Pervasive Computing and Communications Workshops, 2012, pp. 320–325.

[9] A. Caracaş, A. Bernauer, Compiling business process models for sensor networks, in: 2011 Int Conf on Distributed Computing in Sensor Systems and Workshops, 2011, pp. 1–8.

[10] A. Caracaş, T. Kramp, On the expressiveness of bpmn for modeling wireless sensor networks applications, in: Business Process Model and Notation, Springer, 2011, pp. 16–30.

[11] A. Yousfi, C. Bauer, R. Saidi, A. K. Dey, ubpmn: A bpmn extension for modeling ubiquitous business processes, Information and Software Technology (2016) 55–68.

[12] C. T. Sungur, P. Spiess, N. Oertel, O. Kopp, Extending bpmn for wireless sensor networks, in: Conf on Business Informatics, 2013, pp. 109–116.

[13] S. Meyer, A. Ruppen, L. Hilty, The things of the internet of things in bpmn, in: Advanced Information Systems Engineering Workshops, Springer Int Publishing, 2015, pp. 285–297.

[14] P. Valderas, V. Torres, E. Serral, Modelling and executing iot-enhanced business processes through bpmn and microservices, Journal of Systems and Software 184 (2022) 111139.

[15] M. Winter, R. Pryss, T. Probst, J. Baß, M. Reichert, Measuring the cognitive complexity in the comprehension of modular process models, Cogn. Dev. Syst. (2022) 164–180.

[16] M. Winter, H. Neumann, R. Pryss, T. Probst, M. Reichert, Defining gaze patterns for process model literacy - exploring visual routines in process models with diverse mappings, Expert Syst. Appl. (2023) 119217.

[17] C. Ivan, C. Flavio, F. Fabrizio, P. Andrea, R. Barbara, T. Francesco, A systematic literature review on iot-aware business process modeling views, requirements and notations, Journal of Software and Systems Modeling (2022).

[18] V. Torres, E. Serral, P. Valderas, V. Pelechano, P. Grefen, Modeling of iot devices in business processes: A systematic mapping study, in: Conf on Business Informatics, 2020.

Validation of Algorithmic BPMN Layout Classification

Elias Baalmann[1,*], Daniel Lübke[1,2]

[1]Digital Solution Architecture, Hannover, Germany
[2]Leibniz Universität Hannover, FG Software Engineering, Hannover, Germany

Abstract
For many use cases it is handy to clearly and possibly automatically classify the layout direction of BPMN processes, e.g., in empirical research. We want to validate whether our previously proposed classification and algorithm [1] delivers good, reproducible, and helpful results. To accomplish this, we compare the classification algorithm to a previously manually classified large data set of BPMN processes on GitHub. Our results show that the algorithm classifies BPMN layouts similar to manual classification and is suitable for large data sets due to its good run-time characteristics.

Keywords
BPMN, Diagram Layout, Diagram Layout Formalization, Diagram Layout Detection, Flow Layout, Algorithm Validation

1. Introduction

BPMN is the lingua franca for business process modeling and has many use cases. Its main purpose is to convey information between different stakeholders and as such understandability is a key quality feature of BPMN models. Consequently, much research has focused on analysing the impact of different model aspects [2]. This includes the influence of diagram layout [3]. Recently, analysis of large process model repositories like GitHub [4, 5, 6, 7] have become an interesting research direction because large process repositories allow for better statistical results. However, analysing large data sets require much manual work. To address this issue we started to formalize layout directions of BPMN models [1] for improving comparability of studies.In the next step we implemented a tool automating this classification. Within this paper we validate the formalization and the tool implementation by running it against the data set from a previous study [7], comparing the classification results and analyzing the run-time characteristics of our tool.

ZEUS 2023: 15th Central European Workshop on Services and their Composition, February 16–17, 2023, Hannover, Germany
*Corresponding author.
elias.baalmann@digital-solution-architecture.com (E. Baalmann);
daniel.luebke@digital-solution-architecture.com (D. Lübke)
0000-0002-1557-8804 (D. Lübke)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

2. Related Work

One of the first questions that arises in our context is how BPMN diagrams are laid out by practitioners. Effinger et al. [8, p. 400] state that "[i]n BPMN diagrams the flow direction is usually top-to-bottom or left-to right." This statement is empirically validated by Lübke & Wutke [7, p. 52], who found that 79.52% of BPMN diagrams on GitHub are laid out left-to-right. They also identified other layouts, like most prominently, top-down layouts and more complex layouts like multi-line and snake layouts.

A more theoretical approach is taken by Figl & Strembeck. [9, p. 60] who state that "[b]asically, there are four main options for the overall direction: left-to-right, top-to-bottom, bottom-to-top, right-to-left.", i.e., they take all four possible main directions as principal layout directions. However, they have also added that "zigzag models" should be subject to future research, thereby recognizing the use of more complex layouts in practice.

All modeling guidelines we found recommend left-to-right layouts, e.g., the Swiss standard eCH-0158 for eGovernment [10]. Even the BPMN specification itself favors left-to-right modeling [11, p. 42]. Also Corradini et al. [12, p. 49] define a guideline (number 43) that process modelers should make their models long and thin by aligning all edges with a general workflow direction as much as possible.

However, more recently, a study by Lübke et al. [3, p. 127] has shown that the understandability of large diagrams profits from more complex layouts like snake or multi-line layouts to avoid the penalty of scrolling these diagrams on screen. For the case of smaller diagrams, this experiment found a slight advantage for left-to-right layouts in contrast to top-down layouts, affirming Figl & Strembeck's earlier experiment. However, the findings are either minimal (some understandability metrics in the former experiment) or not significant (some metrics in the former experiment and all metrics in the latter experiment).

3. Research Questions

In this paper, we want to answer the following research questions.

RQ1: Does the algorithm proposed by Baalmann & Lübke [1] classify diagrams comparable to manual classification?

RQ2: Is the classification tool suitable to analyze large data sets?

4. Automatic Flow Layout Classification

In his thesis, Baalmann proposes a hierarchy of flow layouts with three levels [13]. The first level describes the base layout: Straight, L, Multi-Line, Stairs, Snake, U, and Z.

The second level differentiates the layouts by orientation. Possible orientations vary for each base layout based on its symmetry. For example, the Straight layout has four orientations: Straight-N, Straight-E, Straight-S, and Straight-W. In this context, compass directions describe the layout direction of the diagram, with E representing left-right, S representing top-down, etc. Table 1 (appendix) lists the possible orientations for each base layout, along with a brief explanation.

In the third level of the hierarchy, minor variations in the layout are examined. These variations typically involve deviations from the general flow of the diagram. However, we will not be considering this level here, as it would go beyond the scope of this paper.

To automatically classify a BPMN diagram based on its flow layout, Baalmann and Lübke use a modular algorithm that reads the BPMN file, splits it into layout paths that connect a start event to an end event which are then processed individually before the results for the paths are finally combined to an overall flow layout of the diagram. To classify a path, it is further split into a vector chain that connects the elements on the path. After simplifying the chain and discretizing the directions of the vectors, a number of regular expressions are used to determine the flow layout of the path.[1]

5. Automatic vs. Manual Classification

In our comparison we will be compare the classification of our tool with the manual classification of 5297 diagrams by Lübke & Wutke [7]. It should be noted that the automation is not intended to replicate the results exactly: diverging results should not necessarily be interpreted as errors on either side. Rather, the goal is to identify reasons for deviations to distinguish the behavior of automated and manual classification.

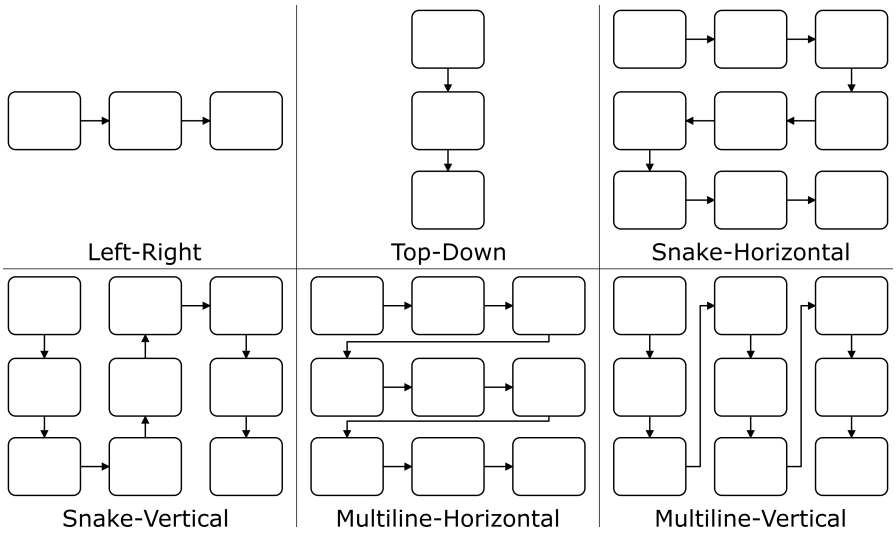


Figure 1: Different flow layouts according to Lübke & Wutke. Figure based on Fig. 2 from [7].

To make comparisons, the flow layouts under consideration must first be matched: The manual classification distinguishes six flow layouts, which are presented in Figure 1. Based on this representation, it is determined that Left-Right corresponds to the Straight-E flow layout. Analogously, Straight-S is the flow layout that is best represented by Top-Down. Since the authors of the manual classification do not specify more precisely which requirements must be met for a particular flow layout, both Snake-ES and Snake-EN are equated with the Snake-Horizontal layout considered by Lübke & Wutke. Following this principle, Snake-Vertical

corresponds to the flow layouts Snake-SE and Snake-NE. In addition, Multiline-Horizontal is compared to Multiline-ES and Multiline-EN, and Multiline-Vertical is compared to Multiline-SE and Multiline-NE. The flow layouts L, Stairs, U, Z, Straight-N, and Straight-W are not considered by Lübke & Wutke.

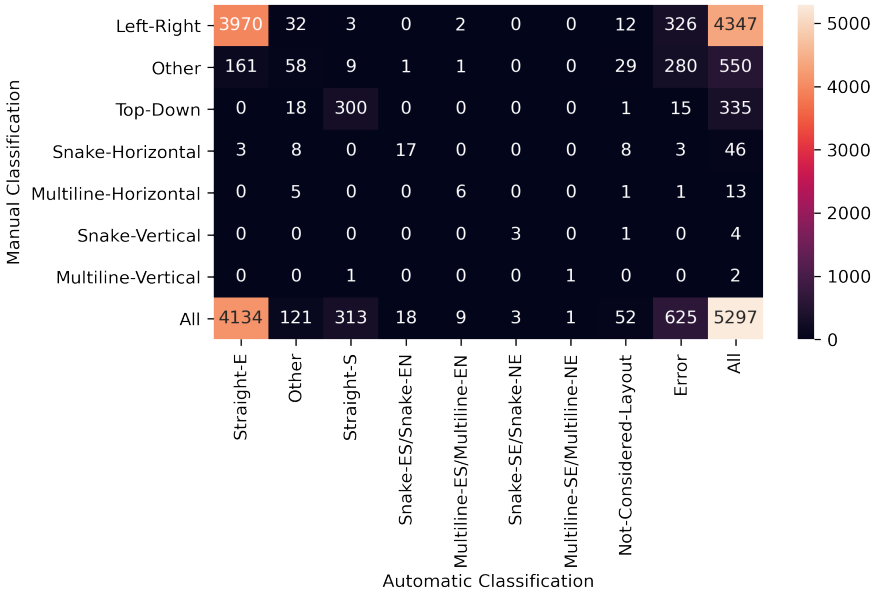


Figure 2: Manual classification of 5297 diagrams compared to automatic classification

Figure 2 shows the distribution of flow layouts per classification for the data set of 5297 diagrams. It is clear that the overwhelming majority of diagrams (manually $4347/5297 \approx 82\%$, automated $4134/5297 \approx 78\%$) are directed from left to right, i.e., classified as Left-Right or Straight-E. It is also notable that the automated variant was unable to classify a large portion of the diagrams ($625/5297 \approx 12\%$) mostly due to invalid BPMN files (for example containing sequence flows between undefined elements), and that a small proportion ($52/5297 \approx 1\%$) of the diagrams were assigned to a flow layout not considered by Lübke and Wutke (Not-Considered-Layout). This raises the question of whether the Not-Considered-Layouts occur frequently enough to be worth considering in future work. However, upon closer examination, it can be seen that in manual classification, only a similarly small proportion ($((46+13+4+2)/5297 \approx 1\%$) of the diagrams were assigned to a flow layout other than Left-Right or Top-Down. Therefore, it is clear that the distribution of the different flow layouts is so uneven that, if flow layouts beyond Straight-E or Straight-S are to be distinguished, a very small number of diagrams must be expected.

Comparing the two classifications, it can be seen that classifications for the straight diagrams (those with the flow layout Left-Right/Straight-E and Top-Down/Straight-S), are very similar. Out of the diagrams manually classified as Left-Right, 91% were automatically classified as Straight-E, and similarly, 90% of those manually classified as Top-Down were automatically classified as Straight-S. However, since 7% and 4% respectively of these diagrams could not

be classified automatically due to errors, it is likely that the agreement is even higher. It is also clear that many (51%) of the diagrams that could not be manually classified as one of the considered flow layouts caused errors in the automated classification. Furthermore, automated classification rarely (about 40%) confirms manual classification when manual classification is Snake-Horizontal, Multiline-Horizontal, or Multiline-Vertical.

Finally it can be seen that the automatic classification in most cases agrees with manual classification when automated classification is able to identify one of the considered flow layouts. The lowest agreement in this sense is for horizontal multi-line variants, at 67%, but it should be noted that many (52%) of the diagrams automatically classified as analyzable (no error) but not classifiable (Other) were manually classifiable.

6. Classification of Large Dataset

To verify that the classification tool is suitable for analyzing large datasets, a large GitHub data set consisting of 48,679 classifiable diagrams is used.

On a desktop PC with an AMD Ryzen 5 3600 CPU, classification of all models takes approximately two hours. However, it should be noted that there are large differences in the time required for each model. The classification of the slowest 10 models took about 99% of the time, while the diagram with the eleventh-longest classification time was classified in less than 30 seconds.

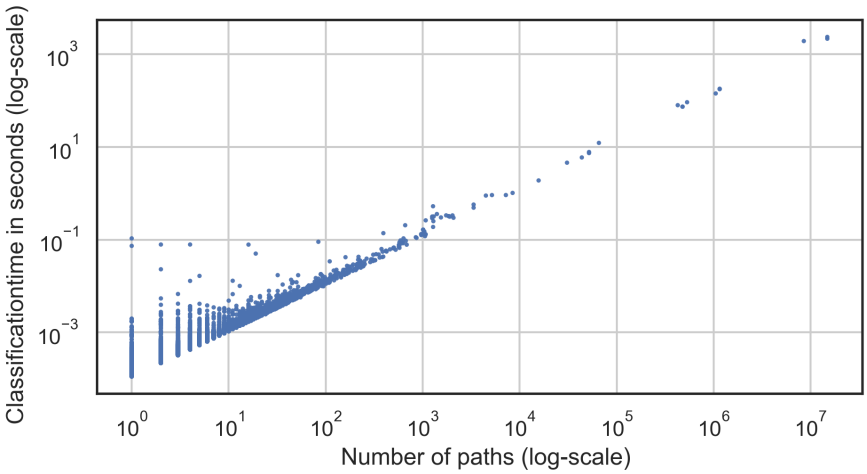


Figure 3: Classification time per model from the GitHub data set. Double-logarithmic representation of the relationship between the number of paths in the model and the classification time.

Figure 3 shows that the run-time of the classification depends on the number of paths in the analyzed model. There seems to be a power law, as the data forms a straight line in the double-logarithmic representation.

To better assess the usefulness of the tool, the statements that can be made about the data set based on automated classification are checked. Figure 4 shows the distribution of basic flow layouts and orientations for the Straight base layout. By far most diagrams have a Straight

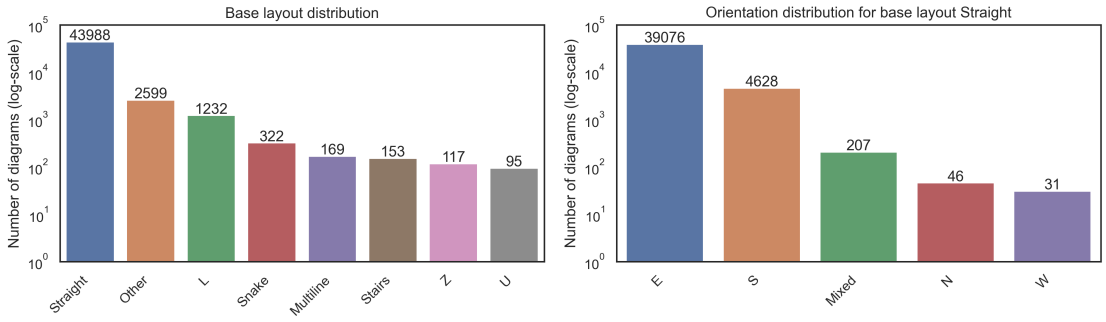


Figure 4: Distribution of flow layouts in large GitHub dataset.

layout (note the logarithmic scale). Furthermore, the orientation E has been assigned to the most diagrams. Specifically, $\frac{39076}{43988} \approx 89\%$ of all diagrams with straight layouts, and therefore $\frac{39076}{48675} \approx 80\%$ of all analyzable diagrams, were classified as Straight-E.

7. Conclusions & Outlook

We could confirm that automated layout classification can be used to analyze large data sets and yields results comparable to manual classification. Besides requiring less effort, automation based on a formalized definition of layouts has additional advantages over manual classification, e.g., it avoids errors due to carelessness and inconsistencies due to subjective perception. However, manual classification currently also has some advantages over automated classification. For example, our tool is not able to complete incomplete diagrams like users with BPMN experience can. In addition, a small number of diagrams are not classified by automation because they do not meet assumptions made in the formalization. It has also been observed that badly laid out diagrams can be better classified manually.

We identified some restrictions of our implementation. For example, models with many paths require a longer processing time possibly making manual inspection more suitable in these cases. However, we encountered hardly any diagrams in our data set, which require a lot of time. If, for example, a time limit of 30 seconds per model had been set, only 11 of the 48679 diagrams would not have been classified. In this case, the total time would have been reduced from about two hours to about five minutes.

Going forward, a way of improving the classification tool would be to reduce the number of not classifiable diagrams by making the implementation less vulnerable to small imperfections in the BPMN files. Another research direction is to extend the validation and get further insights into the problems of the algorithm by using a more diverse data set with diagrams distributed over many different flow layouts. As a side result we could replicate that real-world diagrams are mainly laid out Straight and especially Straight-E (left to right).

We hope that our formalization and tool helps researchers in their empirical studies with BPMN data sets and are open to any cooperation in this regard.

References

[1] E. Baalmann, D. Lübke, Algorithmic Classification of Layouts of BPMN Diagrams, in: CEUR Workshop Proceedings (Ed.), Proceedings of the 14th Central European Workshop on Services and their Composition (ZEUS 2022), volume 3113, 2022, pp. 42–50.

[2] K. Figl, Comprehension of procedural visual business process models, Business & Information Systems Engineering 59 (2017) 41–67.

[3] D. Lübke, M. Ahrens, K. Schneider, Influence of diagram layout and scrolling on understandability of BPMN processes: an eye tracking experiment with BPMN diagrams, Information Technology and Management 22 (2021) 99–131. doi:10.1007/s10799-021-00327-7.

[4] T. Heinze, V. Stefanko, W. Amme, Bpmn in the wild: Bpmn on github. com, in: Proceedings of the 12th ZEUS Workshop on Services and their Composition, CEUR-ws. org, 2020, pp. 26–29.

[5] T. S. Heinze, V. Stefanko, W. Amme, Mining bpmn processes on github for tool validation and development, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2020, pp. 193–208.

[6] J. Türker, M. Völske, T. S. Heinze, Bpmn in the wild: A reprise., in: ZEUS, 2022, pp. 68–75.

[7] D. Lübke, D. Wutke, Analysis of Prevalent BPMN Layout Choices on GitHub, in: CEUR Workshop Proceedings (Ed.), Proceedings of the 13th European Workshop on Services and their Composition (ZEUS 2021), volume 2839, 2021, pp. 46–54.

[8] P. Effinger, M. Siebenhaller, M. Kaufmann, An Interactive Layout Tool for BPMN, in: IEEE Computer Society (Ed.), 2009 IEEE Conference on Commerce and Enterprise Computing, Wien, 2009, pp. 399–406. doi:10.1109/CEC.2009.36.

[9] K. Figl, M. Strembeck, Findings from an experiment on flow direction of business process models, in: J. Kolb, H. Leopold, J. Mendling (Eds.), Enterprise modelling and information systems architectures, Gesellschaft für Informatik e.V, Bonn, 2015, pp. 59–73.

[10] A. Birchler, E. Bosshart, M. Märki, P. Opitz, J. Pauli, B. Rigert, Y. Sandoz, M. Schaffroth, N. Spöcker, C. Tanner, K. Walser, T. Widmer, eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung, WWW: https://www.ech.ch/dokument/fb5725cb-813f-47dc-8283-c04f9311a5b8, 2014.

[11] Object Management Group, Business Process Model and Notation (BPMN), Version 2.0, 2011. URL: https://www.omg.org/spec/BPMN/2.0/PDF.

[12] F. Corradini, A. Ferrari, F. Fornari, S. Gnesi, A. Polini, B. Re, G. O. Spagnolo, Quality assessment strategy: applying business process understandability guidelines for learning, 2015. URL: http://pumax.isti.cnr.it/dfdownloadnew.php?ident=cnr.isti/cnr.isti/2015-TR-034&langver=it&scelta=Metadata.

[13] E. Baalmann, Algorithmische Klassifikation der Layouts von BPMN-Diagrammen, Masterarbeit, Leibniz Universität Hannover, Hannover, 01.04.2022.

A. Appendix

The sources for the automatic classification tool are available via GitHub.

Table 1
Flow Layout Hierarchy

Flow Layout	Directions
Straight-N	bottom-up
Straight-E	left-right
Straight-S	top-down
Straight-W	right-left
L-ES	right, then down
L-WN	left, then up
L-EN	right, then up
L-WS	left, then down
L-SE	down, then right
L-NW	up, then left
L-NE	up, then right
L-SW	down, then left
Multiline-ES	lines left-right, each line below previous line
Multiline-WN	lines right-left, each line above previous line
Multiline-EN	lines left-right, each line above previous line
Multiline-WS	lines right-left, each line below previous line
Multiline-SE	lines top-down, each line right of previous line
Multiline-NW	lines down-top, each line left of previous line
Multiline-NE	lines down-top, each line right of previous line
Multiline-SW	lines top-down, each line left of previous line
Stairs-NE	diagonal bottomleft-topright
Stairs-SE	diagonal topleft-bottomright
Stairs-SW	diagonal topright-bottomleft
Stairs-NW	diagonal bottomright-topleft
Snake-ES	first line left-right, each line below previous line
Snake-WN	first line right-left, each line above previous line
Snake-EN	first line left-right, each line above previous line
Snake-WS	first line right-left, each line below previous line
Snake-SE	first line top-down, each line right of previous line
Snake-NW	first line down-top, each line left of previous line
Snake-NE	first line down-top, each line right of previous line
Snake-SW	first line top-down, each line left of previous line
U-ES	left-right, then top-down, then right-left
U-WN	right-left, then bottom-up, then left-right
U-EN	left-right, then bottom-up, then right-left
U-WS	right-left, then top-down, then left-right
U-SE	top-down, then left-right, then bottom-up
U-NW	bottom-up, then right-left, then top-down
U-NE	bottom-up, then left-right, then top-down
U-SW	top-down, then right-left, then bottom-up
Z-ES	left-right, then top-down, then left-right
Z-WN	right-left, then bottom-up, then right-left
Z-EN	left-right, then bottom-up, then left-right
Z-WS	right-left, then top-down, then right-left
Z-SE	top-down, then left-right, then top-down
Z-NW	bottom-up, then right-left, then bottom-up
Z-NE	bottom-up, then left-right, then bottom-up
Z-SW	top-down, then right-left, then top-down

Execution Semantics of Process Models with Data

Maximilian König[1]

[1]Hasso Plattner Institute, Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam

Abstract

Data is one of today's most important currencies. Thus, maintaining an overview of its creation, usage, and manipulation within an organization is of utmost importance. While this fact has been recognized in the Business Process Management (BPM) community in general, its subfield of process modeling has not attributed attention to that for a long time. Extensive research has been conducted on the logical and temporal order of process steps, also called the control flow. While doing so, the impact of data, e.g., that certain tasks require specific information to be executed, has been largely neglected. Even with the extension of some process modeling languages to incorporate data concepts, formal semantics of these concepts, that enable automated analysis and enactment, are often not present or underspecified. Therefore, this paper motivates the definition of a new, more holistic semantics for data concepts in BPMN. This semantics is then to be used as a foundation to adapt existing and define novel verification, compliance, and consistency checking methods regarding the data and data flow of processes.

Keywords

Data in Processes, BPMN, Translational Semantics, Petri nets

1. Introduction

For an organization to thrive in a fast-paced, competitive environment, it must constantly monitor its business processes and the data required for and manipulated by their execution. That requires a thorough documentation of these processes and the information involved in them. A means to achieve that are process models describing the required tasks and their data pre- and postconditions. However, due to the size and number of processes in place within an organization, manually managing these process models and ensuring their consistency, correctness, and compliance, especially when undergoing change, is very challenging and error-prone. Hence, automation is desirable, but requires thorough formalization of the involved concepts, i.e., the definition of a concise execution semantics. Looking at the Business Process Modeling and Notation (BPMN) [1], the most widely adopted process modeling language [2], multiple approaches exist defining a concise semantics for its control flow concepts. However, there is currently no formalization that covers all of its data concepts, which prevents automated analyses of process data flow. Therefore, this paper motivates the introduction of a new execution semantics defined through the mapping of BPMN data concepts to Petri net constructs that can serve as a basis for the automated analysis of the data flow of process models.

ZEUS2023: 15th Central European Workshop on Services and their Composition, February 16–17, 2023, Hannover, Germany

maximilian.koenig@student.hpi.de (M. König)

0000-0002-2244-1179 (M. König)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

In the remainder of this paper, BPMN data concepts are briefly introduced and related work on formal semantics for BPMN is discussed before the new mapping and its potential application areas are outlined.

2. Background and Related Work

Since version 2.0, the BPMN standard [1] includes a number of semantically meaningful concepts regarding the data flow in process models. A minimal example containing a subset of them is shown in Figure 1. *Data objects* (document shapes) are an abstract representation of the data used in a process. They can have *data states* (denoted in squared brackets) assigned to them. Connections to activities indicate that the respective data object is required as input or produced as output. In- and outgoing data objects can be clustered in *input and output sets* (I/O sets, denoted using, e.g., *I1* and *O1*) representing sufficient data enablement or termination conditions. For example, a *Claim* in state *[received]* alone is sufficient to start *Assess Insurance Claim*. Input output specifications (I/O specs, denoted using the BPMN annotation element) define, which data may be produced by an activity given it started with a certain input set. In the example, if an *Insuree Scam History* data object exists (*I2*), the risk assessment always produces a *Risk* in state *[high]* and a *Second Assessment* in state *required* (*O2*). Finally, the BPMN standard employs the rule that only a single data object instance may exist per process instance, unless specified as being multi-instance.

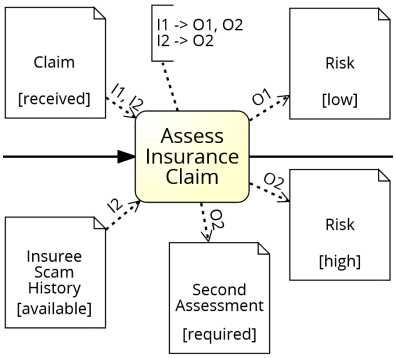


Figure 1: BPMN data concepts

While the BPMN standard provides a textual description of the semantics of these concepts, a thorough formalization is currently missing. In the past, research has been conducted on defining such a formalization through translational semantics, i.e., creating a mapping of BPMN concepts to those of another, well-formalized modeling language [3]. Target languages comprise, inter alia, process algebras [4], WS-BPEL [5], Event-B [6], and, most prominently, Petri nets [7, 8, 9, 10, 11, 12, 13]. However, most approaches solely consider the control flow and disregard the data perspective. Even the ones considering the data perspective only define semantics for subsets of the above-mentioned data concepts. Table 1 shows an overview of a selection of works introducing a translational semantics of BPMN to (colored) Petri nets, and the data-related BPMN concepts they cover. In addition to the concepts introduced in the standard, the table includes *Data Locking*, which represents a mechanism to prevent concurrent write access and inconsistent read access while the data may concurrently be modified as proposed in [9]. The overview shows that, while all aspects are covered by at least one approach, no mapping yet covers all aspects.

Next to the extension of an activity-centric process modeling language with data concepts, related work proposed a variety of different approaches to the representation of data in business processes. Two frameworks have been introduced to evaluate and compare approaches regarding their incorporation of data in business processes [14, 15]. The covered spectrum ranges

from extensions to activity-centric approaches such as BPMN [9] through case management, e.g., fragment-based Case Management [16], to artifact-centric, e.g., [17], and object-centric approaches, e.g., [18]. However, most of these approaches require the learning of a new process modeling language, which constitutes a major hurdle to their adoption. In contrast, BPMN already is the de facto industry standard. Hence, providing the precision the BPMN standard currently lacks regarding its data concepts may lower the threshold to incorporate the data perspective in existing BPMN process models and will therefore be the focus of this work.

Table 1
Coverage of BPMN data concepts in related works mapping BPMN to Petri nets. Each aspect is either covered (✓), partially covered ([✓]), or not covered (–). SI abbreviates *Single-Instance enforcement*.

Publication	Data Objects	Data States	I/O Sets	I/O Spec	Data Locking	SI
Dijkman et al. [7]	–	–	–	–	–	–
Meyer [9]	✓	[✓]	–	–	✓	✓
Awad et al. [10]	✓	✓	–	–	–	–
Stackelberg et al. [11]	✓	–	✓	–	–	✓
Ramadan et al. [13]	✓	[✓]	✓	✓	–	–
Dechsupa et al. [12]	✓	–	–	–	–	–

3. Contribution

To close the gap of a missing holistic semantics for data concepts in BPMN, as outlined in the preceding section, this paper proposes the introduction of a new translational semantics for BPMN using Petri nets. The mapping should cover data objects, data states, input and output sets, input and output specifications, multi-instance and single-instance behavior, a data locking mechanism, and the combination of these concepts with tasks, subprocesses, and (boundary) events adhering to the restrictions provided in the BPMN standard. To allow a focus on the data flow, the mapping rules for BPMN control flow elements introduced by Dijkman et al. [7] will be extended. The result should be a formal algorithm based on which properties of the derived nets such as the enforcement of a single data object instance per process instance, data locking as defined by Meyer [9], and the correct translation of BPMN concepts can be proven.

The Petri nets resulting from the algorithm's application will then serve as a foundation for a set of analyses. First, a notion of data flow soundness should be defined, ensuring the absence of data deadlocks and proper termination of the processes. Next to that, different categories of data anomalies have been defined in literature [11, 19, 20]. Their application as well as a potential extension of them based on the introduced formalism would allow for a more thorough analysis of processes' data flow. For example, lost data (data objects being written multiple times without being read in between), missing data (required data in a certain process state that is not available), and redundant data (data that is written but never read) can thereby be detected. In addition, a recent literature review revealed a lack of data compliance checking methods for processes [21]. In that context, it would be interesting to see which of their constraint patterns concerning data will be detectable in the derived Petri nets.

4. Conclusion

This position paper motivates the need for a new translational semantics thoroughly covering BPMN's data concepts. The lack of such a semantics hinders the capitalization on process models' capability to provide and maintain an overview of the data flow within an organization. The mapping of BPMN concepts to Petri nets should then serve as the foundation for automated data flow analyses regarding correctness, consistency with other processes, and compliance to the process environment's regulations.

References

[1] OMG, Business Process Model and Notation (BPMN), Version 2.0.2, Technical Report, Object Management Group, 2014. https://www.omg.org/spec/BPMN/2.0.2/PDF.

[2] M. Dumas, D. Pfahl, Modeling software processes using BPMN: When and when not?, in: Managing Software Process Evolution, Springer, 2016, pp. 165–183.

[3] A. H. M. ter Hofstede, H. A. Proper, How to Formalize it?: Formalization Principles for Information System Development Methods, Inf. Softw. Technol. 40 (1998) 519–540. doi:10.1016/S0950-5849(98)00078-0.

[4] P. Y. H. Wong, J. Gibbons, A Process Semantics for BPMN, in: S. Liu, T. S. E. Maibaum, K. Araki (Eds.), Formal Methods and Software Engineering, 10th International Conference on Formal Engineering Methods, ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008. Proceedings, volume 5256 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 355–374. doi:10.1007/978-3-540-88194-0_22.

[5] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, W. M. P. van der Aalst, From BPMN Process Models to BPEL Web Services, in: 2006 IEEE International Conference on Web Services (ICWS 2006), 18-22 September 2006, Chicago, Illinois, USA, IEEE Computer Society, 2006, pp. 285–292. doi:10.1109/ICWS.2006.67.

[6] J. W. Bryans, W. Wei, Formal analysis of BPMN models using event-b, in: S. Kowalewski, M. Roveri (Eds.), Formal Methods for Industrial Critical Systems - 15th International Workshop, FMICS 2010, Antwerp, Belgium, September 20-21, 2010. Proceedings, volume 6371 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 33–49. doi:10.1007/978-3-642-15898-8_3.

[7] R. M. Dijkman, M. Dumas, C. Ouyang, Semantics and Analysis of Business Process Models in BPMN, Inf. Softw. Technol. 50 (2008) 1281–1294. doi:10.1016/j.infsof.2008.02.006.

[8] I. Raedts, M. Petkovic, Y. S. Usenko, J. M. E. M. van der Werf, J. F. Groote, L. J. Somers, Transformation of BPMN Models for Behaviour Analysis, in: J. C. Augusto, J. Barjis, U. Ultes-Nitsche (Eds.), Modelling, Simulation, Verification and Validation of Enterprise Information Systems, Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS 2007, Funchal, Madeira, Portugal, June 2007, INSTICC PRESS, 2007, pp. 126–137.

[9] A. Meyer, Data Perspective in Business Process Management, PhD Thesis, Universität Potsdam, 2015.

[10] A. Awad, G. Decker, N. Lohmann, Diagnosing and Repairing Data Anomalies in Process Models, in: S. Rinderle-Ma, S. W. Sadiq, F. Leymann (Eds.), Business Process Management Workshops, BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers, volume 43 of *Lecture Notes in Business Information Processing*, Springer, 2009, pp. 5–16. doi:10.1007/978-3-642-12186-9_2.

[11] S. von Stackelberg, S. Putze, J. Mülle, K. Böhm, Detecting Data-Flow Errors in BPMN 2.0, Open Journal of Information Systems (OJIS) 1 (2014) 1–19. URL: http://nbn-resolving.de/urn:nbn:de:101:1-2017052611934.

[12] C. Dechsupa, W. Vatanawood, A. Thongtak, Hierarchical Verification for the BPMN Design Model Using State Space Analysis, IEEE Access 7 (2019) 16795–16815. doi:10.1109/ACCESS.2019.2892958.

[13] M. Ramadan, H. G. Elmongui, R. Hassan, BPMN formalisation using coloured petri nets, in: Proceedings of the 2nd GSTF annual international conference on software engineering & applications (SEA 2011), 2011, pp. 83–90.

[14] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, M. Reichert, DALEC: a framework for the systematic evaluation of data-centric approaches to process management software, Softw. Syst. Model. 18 (2019) 2679–2716. URL: https://doi.org/10.1007/s10270-018-0695-0. doi:10.1007/s10270-018-0695-0.

[15] A. Meyer, S. Smirnov, M. Weske, Data in Business Processes, EMISA Forum 31 (2011) 5–31.

[16] M. Hewelt, M. Weske, A Hybrid Approach for Flexible Case Modeling and Execution, in: M. L. Rosa, P. Loos, O. Pastor (Eds.), Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings, volume 260 of *Lecture Notes in Business Information Processing*, Springer, 2016, pp. 38–54. doi:10.1007/978-3-319-45468-9_3.

[17] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. F. T. H. III, S. Hobson, M. H. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, R. Vaculín, Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles, in: M. Bravetti, T. Bultan (Eds.), Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers, volume 6551 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 1–24. doi:10.1007/978-3-642-19589-1_1.

[18] V. Künzle, M. Reichert, PHILharmonicFlows: towards a framework for object-aware process management, J. Softw. Maintenance Res. Pract. 23 (2011) 205–244. doi:10.1002/smr.524.

[19] S. W. Sadiq, M. E. Orlowska, W. Sadiq, C. Foulger, Data Flow and Validation in Workflow Modelling, in: K. Schewe, H. E. Williams (Eds.), Database Technologies 2004, Proceedings of the Fifteenth Australasian Database Conference, ADC 2004, Dunedin, New Zealand, 18-22 January 2004, volume 27 of *CRPIT*, Australian Computer Society, 2004, pp. 207–214. URL: http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV27Sadiq.html.

[20] S. X. Sun, J. L. Zhao, J. F. N. Jr., O. R. L. Sheng, Formulating the Data-Flow Perspective for Business Process Management, Inf. Syst. Res. 17 (2006) 374–391. doi:10.1287/isre.1060.0105.

[21] T. Voglhofer, S. Rinderle-Ma, Collection and Elicitation of Business Process Compliance Patterns with Focus on Data Aspects, Bus. Inf. Syst. Eng. 62 (2020) 361–377. doi:10.1007/s12599-019-00594-3.

Discovering Process Models of Different Granularity from Legacy Software Systems

Marius Breitmayer[1,*,†], Lisa Arnold[1,†], Stephan La Rocca[2,†] and Manfred Reichert[1,†]

[1]*Institute of Databases and Information Systems, Ulm University, Germany*
[2]*PITSS GmbH Stuttgart, Germany*

Abstract

Processes models shall enable a better understanding and management of business processes as well as the information systems implementing these processes. Usually, different stakeholders of various enterprise levels are interested in process models which raises specific requirements concerning process model abstraction. While business managers are interested in high-level (i.e., abstract) process views, process participants need more fine-grained process views during process enactment. This also applies to many legacy software systems that implement business processes, but were originally not designed to provide process model details. In this paper, we present an approach for preprocessing event logs obtained from legacy software systems such that process models of different granularity levels can be discovered from them.

Keywords

process mining, event log, preprocessing, process model abstraction

1. Introduction

The organizational structure of an enterprise may be considered as a set of methods through which the organization is logically divided into distinct sets of business functions. The latter need to be harmonized across multiple granularity levels (e.g., different hierarchy levels of the organisational structure) to achieve enterprise goals [1, 2]. Moreover, the required tasks are often structured in terms of process models, which are used by various stakeholders from different enterprise hierarchy levels (e.g., managers, executives, or employees). Usually, these stakeholders have different expectations concerning the level of granularity in which a process model shall be displayed. For example, specific data required in the context of a particular process task might not be relevant for managers, but are crucial for process participants to correctly perform their tasks at runtime. Consequently, process models of different granularity levels need to be discovered to meet those expectations.

Process mining and process discovery, respectively, leverage recorded audit or workflow data (i.e., event logs) to reconstruct the business processes implemented by enterprise information

15th Central European Workshop on Services and their Composition, February 16–17, 2023, Hannover, Germany
*Corresponding author.
†These authors contributed equally.
marius.breitmayer@uni-ulm.de (M. Breitmayer); lisa.arnold@uni-ulm.de (L. Arnold); slarocca@pitss.com (S. La Rocca); manfred.reichert@uni-ulm.de (M. Reichert)
0000-0003-1572-4573 (M. Breitmayer); 0000-0002-2358-2571 (L. Arnold); 0000-0003-2536-4153 (M. Reichert)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

systems [3, 4]. In general, the discovered process models are strongly correlated with both the existence of suitable event logs and their quality [5]. Although there exist mature process discovery algorithms, which are able to deal with noise [6] and incompleteness [7], in the context of legacy software applications their application is still limited, as process-related event logs are usually not readily available in these systems [8].

In previous work [9, 10], we have presented a possible solution to this challenge. Basically, our approach observes the interactions of process participants with the legacy software system and records these interactions in a fine-grained way, which shall also allow obtaining event logs that are suitable for process discovery algorithms. This paper presents an approach to abstract the fine-grained information recorded in corresponding event logs to enable the discovery of process models suitable of different granularity levels.

The remainder of this paper is structured as follows: Section 2 describes the proposed solution and shows how it allows creating event logs constituting the basis for discovering process models of different levels of granularity. Section 3 describes the discovery algorithm applied to these logs as well as the proposed hierarchy levels for resulting process models. In Section 4, we evaluate the approach using a real-world event log. Section 5 discusses related work. Finally, Section 6 provides a summary and outlook.

2. Solution Approach

In the context of legacy software systems, we define a process as a sequence of related interactions the users have with the system [10]. Consequently, an event log derived from a legacy software system is described by entries of which each corresponds to an individual user interaction. In general, user interactions are driven by the structure and layout of the user forms (e.g., to check an invoice or to process an order) of the legacy software system. In an Oracle legacy system, for example, a graphical user interface (GUI) is located on a canvas (see L3 in Fig. 1). The latter, in turn, may comprise several screens or canvases (see L2 in Fig. 1) which again contain form fields and buttons (see L1 in Fig. 1). Consequently, the events recorded in an event log contain information like the respective form field filled or button clicked by a user as well as the form or screen in which the user interaction took place. An event log obtained from an Oracle legacy software system, therefore, comprises event data, which are characterized by a user story (i.e., a case identifier), timestamps (e.g., entering or leaving a field), additional information based on user specification, and the actual user interaction (e.g., button 'X' clicked or field 'Y' filled). A user interaction is structured in the way *ScreenName.EventName*, and examples include ORDERS.DATE_CONTROL_BLOCK.MONTH_PLUS1.WHEN-BUTTON-PRESSED (i.e., in the order screen a button to increase a date by one month was pressed) and ORDERS.MAIN_CANVAS.BUTTON_SAVE.WHEN_BUTTON_PRESSED (i.e., in the main canvas of the order screen the save button was pressed). In other words, the event log documents user interactions at the finest granularity level. Discovering process models based on such fine-grained event logs, however, might lead to complex process models that are hard to read and understand [11]. To tackle this challenge, we preprocess the event log stored form legacy software systems, while considering hierarchies and the event log structure. Note that this shall allow for the discovery of multiple process models with different granularity based on the same initial event log.

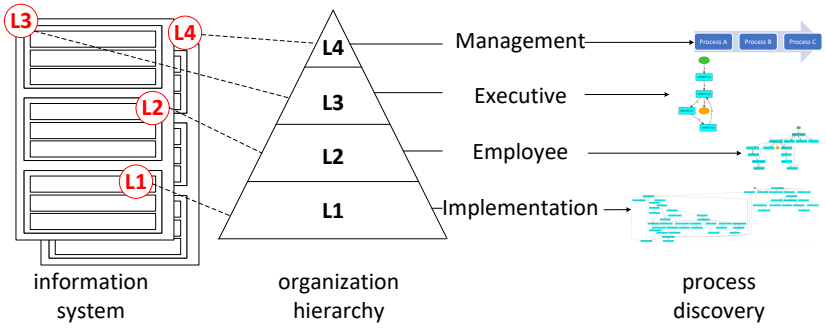


Figure 1: Proposed approach for event log preprocessing with hierarchies

2.1. Event Log Hierarchies

The different stakeholders intended in process models require different levels of abstraction. By preprocessing event logs accordingly, we are also able to discover process models of different granularity levels. For this purpose, we extract corresponding hierarchical information from the individual user interactions taking the structure of the form elements the respective user has interacted with into account. The resulting event log and event log entries respectively allow differentiating between four different granularity levels:

The **Management Level** shall refer to multiple processes and show how these processes are connected and coordinated (see L4 in Fig. 1).

The **Executive Level** deals with a high-level view of the processes – it solely considers the events related to the completion of forms during process execution (see L3 in Fig. 1). Example: ORDERS

The **Employee Level** considers more detailed information on the form screens required to complete a process (see L2 in Fig. 1). Example: ORDERS.MAIN_CANVAS

The **Implementation Level** covers the most fine-grained event log representation. It also includes specific form fields and buttons (see L1 in Fig. 1). Example: ORDERS.MAIN_CANVAS.BUTTON_SAVE.WHEN_BUTTON_PRESSED

When abstracting the event log to a higher granularity level (e.g., from the employee level to the executive level), we remove consecutive identical event log entries (e.g., multiple interactions in the same form are recorded). This reduces the complexity of process models of higher hierarchical levels.

Finally, the fine-grained event logs enable a mapping between the event label (e.g., BUTTON_SAVE.WHEN_BUTTON_PRESSED) and a less technical representation of the event label (e.g., Button Save Pressed). A less technical language representation of labels fosters the comprehension of the fine-grained process model, and, therefore, increases the comprehension of discovered process models.

Table 1
Domain Expert Recognition (N=13) [10].

	Inductive (Tree)	Inductive (BPMN)	DFG	Heuristic (thold=0.75)	Heuristic (thold=0.9)	Heuristic (thold=0.95)
Mean (SD)	3.08 (1.07)	3.08 (0.73)	2.31 (1.2)	4.15 (0.77)	4.46 (0.63)	3.38 (1.27)

3. Process Discovery

3.1. Algorithm Selection

Prior to applying process discovery algorithms to the preprocessed event logs, we conducted a Delphi study with domain experts in order to identify which process discovery algorithm yields the most appropriate results [10]. Study participants (N=13) were asked to evaluate to which degree they could recognize the legacy software system based on the resulting process models on a scale from 1 (not at all) to 5 (completely). Table 1 presents the results.

The process models generated by the Heuristic Miner yielded the best results regarding recognition. Therefore, we apply the Heuristic Miner for the discovery of process models for individual hierarchy levels [12].

3.2. Granularity Levels

The **implementation level (L1)** reflects to the most fine-grained representation of the recorded user interactions. It represents the individual interactions the user(s) had with the legacy software system. Corresponding process models foster the migration of a legacy software systems to modern technologies as the event log documents all user interactions that occurred at runtime, including exceptions that occurred in the legacy software system.

The **employee level (L2)** deals with the screens and canvases of a form filled by users when interacting with the legacy software system. Usually, corresponding screens and canvases contain form fields dealing with the same topic, e.g., contact data of a supplier or the items of an order. Consequently, the discovery of processes on the employee-level provides employees with an overview of process-relevant blocks that need to be completed in order to execute a process. This, in turn, enables a better understanding as well as acceptance of executed processes for employees.

While the employee level is concerned with the screens and canvases of a form, the **executive level (L3)** further considers the order in which users usually interact with the forms. Process models of the executive level document in which way the users navigate through the variety of forms (i.e., in which order different forms are filled in). This allows for an abstracted view on the processes implemented in the legacy system, while at the same time enabling a high-level view on the forms to be completed during process execution.

The **management level (L4)**, does not relate different forms as it relates multiple processes. A model discovered on this level shows how different processes are executed in order to achieve a goal. In this setting, different processes implemented by the legacy software system are related to each of them and, consequently, discovered.

Table 2
Comparison of the Resulting Process Models and their Granularity Levels

	Implementation Level	Employee Level	Executive Level
Number of Activities	52	15	4
Reduction (%)	0.00%	-71.15%	-92.31%

4. Evaluation

We applied the presented approach for preprocessing event logs to an event log we generated through the dedicated recording of the sessions a particular user had with an Oracle legacy software system in an industrial setting [9, 10]. The user could provide supplementary information for the recorded business process (e.g., description and context of the process). In total, the considered legacy software system comprises 589 database tables with 9977 columns. More than 8000 different database statements (60712 statements in total) were implemented in more than 5 million lines of code. Finally, the legacy software system comprises 1285 forms and 6243 different screens.

We preprocessed the event log to provide the information specific to the hierarchical levels introduced in Section. 2. Subsequently, we applied the heuristic miner to discover the process models for each hierarchy level. Note that we used the heuristic miner with default configuration (i.e., dependency_threshold = 0.5, and_threshold = 0.65, loop_two_threshold = 0.5) to discover the process models. Fine-tuning of these parameters might further improve the quality of generated process models. Figures 2 - 4 depict the resulting process models for hierarchy levels L1 to L3. As the collected event log solely contains the interactions of a single process (i.e., a single user story), we were unable to discover a process model for the management level. The event log as well as the process models are provided in an anonymous version[1]. When juxtaposing the process models discovered by the heuristic miner, the different levels of granularity for each organizational level become evident. In case of the three process models discovered from the legacy system event log (see Figs. 2 - 4), the number of activities are described in Table 2. In the given event log, the overall number of activities was reduced by 71.15% (implementation vs. employee level), and by 92.31% (implementation vs. executive level). A reduction of 73.33% in terms of the number of activities can be observed when comparing employee and executive level.

The size (i.e., the number of elements) of a process model does effect both its understandability [13] as well as the likelihood of errors [14, 15]. According to the 7 Process Modelling Guidelines [16], larger process models are more difficult to understand, and have a higher error probability. In our approach, the number of discovered activities decreases for higher hierarchy levels. Consequently, our approach for preprocessing and discovering process models from legacy software systems is able to provide suitable process models for various stakeholders as the granularity decreases for higher hierarchy levels. To be more precise, process models suitable for of higher level hierarchies (i.e., L2-L4) allow for the abstraction of fine-grained event data.

[1] https://cloudstore.uni-ulm.de/s/iHyJtA9riioyJEK

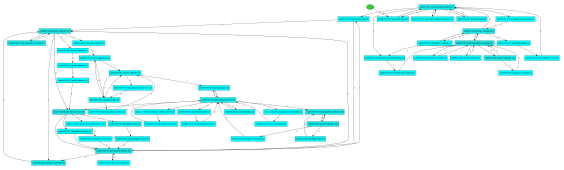


Figure 2: Implementation level

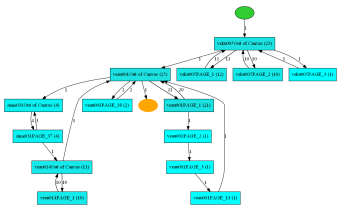


Figure 3: Employee level

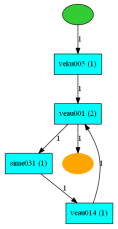


Figure 4: Executive level

5. Related Work

This work is related to the research areas of event log preprocessing [17] and event log abstraction [11] in process mining [3].

Event log preprocessing is concerned with tasks that substantially improve the performance of process mining algorithms by detecting and removing noise as well as traces and activities that contain undesired behavior [17]. Preprocessing is either concerned with transformation or detection techniques. Transformation techniques filter the event log based on a likelihood and, consequently, remove unlikely event log entries [18], or they delete erroneous event log entries [19]. Other approaches consider temporal aspects during event log preprocessing [20]. Detection techniques try to identify those events that are problematic for the quality of an event log based on patterns and clustering techniques [17]. Approaches for event clustering usually use some sort of internal representation for event logs, derive corresponding clusters, and use them to aggregate the event log [21, 22]. As opposed to existing approaches for event log transformation and event log detection, our approach leverages information from the structure of an information system (i.e., how users interact with the system and how corresponding user forms are organized), to provide process models suitable for readers of different granularities.

Regarding event log abstraction, supervised techniques use time intervals [23], manual mappings [24], views on the process model [25, 26] or reference models [27] to abstract behavior in the event log, whereas unsupervised techniques aim to identify re-occurring patterns [28, 29]. In contrast, our approach leverages domain knowledge about the event log automatically derived from the structure of the legacy software system to aggregate event log entries accordingly.

6. Conclusion and Outlook

This paper presents an approach to leverage information from (legacy) software systems to preprocess event data, while considering various hierarchies to provide suitable process models for different stakeholders. This not only enables non-domain experts to better understand a legacy software system, but the discovered process models also serve as process documentation for the legacy systems, facilitating software migration projects. Additionally, process models for higher organizational levels (e.g., employee, executive or management) are less complex and, therefore, more suitable for humans to understand especially. This is helpful for stakeholders to understand the process models, and to identify improvement potential. In future work, we apply the presented approach to additional event logs obtained from legacy software systems, as well as event logs that are not collected from a dedicated recording.

References

[1] H. Mintzberg, The structuring of organizations: a synthesis of the research, Prentice-Hall Englewood Cliffs, N.J, 1979.

[2] Q. Tran, Y. Tian, Organizational structure: Influencing factors and impact on a firm, American Journal of Industrial and Business Management 03No.02 (2013) 8.

[3] W. M. P. van der Aalst, Process Mining: Data Science in Action, Springer, 2016.

[4] W. M. P. van der Aalst, Process discovery: Capturing the invisible, IEEE Computational Intelligence Magazine 5 (2010) 28–41.

[5] W. M. P. van der Aalst, et al., Process mining manifesto, in: Int'l Conf on BPM'11, 2011, pp. 169–194.

[6] A. Weijters, J. Ribeiro, Flexible heuristics miner (fhm), in: Symp on CIDM'11, 2011, pp. 310–317.

[7] A. K. A. de Medeiros, A. J. Weijters, W. M. P. van der Aalst, Genetic process mining: an experimental evaluation, Data Mining and Knowledge Discovery 14 (2007) 245–304.

[8] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods, Springer International Publishing, Cham, 2019, pp. 3–25.

[9] M. Breitmayer, L. Arnold, M. Reichert, Towards retrograde process analysis in running legacy applications, in: 14th Central European Workshop on Services and their Composition (ZEUS 2022), number 3113 in CEUR Workshop Proceedings, 2022.

[10] M. Breitmayer, L. Arnold, S. L. Rocca, M. Reichert, Deriving event logs from legacy software systems, in: 4th International Conference on Process Mining (ICPM 2022), ICPM 2022 Workshops, Springer Nature Switzerland AG, 2022.

[11] S. J. van Zelst, F. Mannhardt, M. de Leoni, A. Koschmider, Event abstraction in process mining: literature review and taxonomy, Granular Computing 6 (2021) 719–736.

[12] A. J. M. M. Weijters, W. M. P. van der Aalst, A. K. A. de Medeiros, Process mining with the heuristicsminer algorithm, 2006.

[13] J. Mendling, H. A. Reijers, J. Cardoso, What makes process models understandable?, in: Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 48–63.

[14] J. Mendling, G. Neumann, W. M. P. van der Aalst, Understanding the occurrence of errors in process models based on metrics, in: On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 113–130.

[15] J. Mendling, H. Verbeek, B. van Dongen, W. van der Aalst, G. Neumann, Detection and prediction of errors in epcs of the sap reference model, Data Knowledge Engineering 64 (2008) 312–329. Fourth International Conference on Business Process Management (BPM 2006) 8th International Conference on Enterprise Information Systems (ICEIS' 2006).

[16] J. Mendling, H. A. Reijers, W. M. P. van der Aalst, Seven process modeling guidelines (7pmg), Inf. Softw. Technol. 52 (2010) 127–136.

[17] H. M. Marin-Castro, E. Tello-Leal, Event log preprocessing for process mining: A review, Applied Sciences 11 (2021).

[18] R. Conforti, M. L. Rosa, A. H. t. Hofstede, Filtering out infrequent behavior from business

process event logs, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 300–314.

[19] N. Tax, N. Sidorova, W. M. P. van der Aalst, Discovering more precise process models from event logs by filtering out chaotic activities, Journal of Intelligent Information Systems 52 (2019) 107–139.

[20] S. Song, Y. Cao, J. Wang, Cleaning timestamps with temporal constraints 9 (2016) 708–719.

[21] R. Jagadeesh Chandra Bose, W. Aalst, van der, Context aware trace clustering : towards improving process mining results, in: Proceedings of the Ninth SIAM International Conference on Data Mining (SDM 2009, Sparks NV, USA, April 30-May 2, 2009), Society for Industrial and Applied Mathematics (SIAM), 2009, pp. 401–412.

[22] M. Boltenhagen, T. Chatain, J. Carmona, Generalized alignment-based trace clustering of process behavior, in: Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, 2019, pp. 237–257.

[23] M. L. van Eck, N. Sidorova, W. M. P. van der Aalst, Enabling process mining on sensor data from smart products, in: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), 2016, pp. 1–12.

[24] A. Begicheva, I. Lomazova, Discovering high-level process models from event logs, Modeling and Analysis of Information Systems 24 (2017) 125–140.

[25] J. Kolb, M. Reichert, A flexible approach for abstracting and personalizing large business process models, Applied Computing Review 13 (2013) 6–17.

[26] R. Bobrik, M. Reichert, T. Bauer, View-based process visualization, in: 5th Int'l Conf. on Business Process Management (BPM'07), number 4714 in LNCS, Springer, 2007, pp. 88–95.

[27] M. de Leoni, S. Dündar, Event-log abstraction using batch session identification and clustering, in: Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC 2020, 2020, pp. 36–44.

[28] R. P. Jagadeesh Chandra Bose, W. M. P. van der Aalst, Abstractions in process mining: A taxonomy of patterns, in: Business Process Management, 2009, pp. 159–175.

[29] C. W. Günther, A. Rozinat, W. M. P. van der Aalst, Activity mining by global trace segmentation, in: Business Process Management Workshops, 2010, pp. 128–139.

Towards Progress Determination in Dynamically Evolving Large Process Structures

Lisa Arnold[1,*], Marius Breitmayer[1] and Manfred Reichert[1]

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

Abstract

Real-time business process monitoring shall enable the early detection of problems and errors, that might occur during process execution. In turn, this allows counteracting possible delays, breakdowns, or defects at an early stage and, thus, increasing the economic efficiency and efficacy of the business processes. In this context, the calculation of the progress of a dynamically evolving process structure provides the basis for advanced monitoring functions related to resource management, risk management, alarm triggering, and error warnings. This position paper discusses the fundamental challenges of determining the progress of multiple, interacting business objects in dynamically evolving object-centric business process during run time. In particularly we introduce five (sub-)research questions that need to be investigated in this context.

Keywords

object-centric business process, process monitoring, progress determination, online/real-time monitoring

1. Introduction

Business process monitoring constitutes a key element for companies to control, optimise, and evolve their business process. Moreover, it allows for an early discovery of process errors or other problems that might occur during process execution. In general, monitoring large, and dynamically evolving business process structures that allow for a high flexibility as well as dynamic changes during run time, (e.g., to dynamically add or delete business objects and the corresponding lifecycle processes) is a challenging task [1, 2, 3]. In this context, determining the progress of large and dynamically evolving process structures is a fundamental task of any business process monitoring composing comprises a wide range of open issues for its calculation. In this paper, we discuss research questions that become relevant in the context of determining the progress in object-centric business processes with multiple, interacting business objects and corresponding lifecycle processes. Both, the progress of the lifecycle processes of single business objects and further the progress of the overall relational process structure instance comprising a potentially large number of lifecycle processes need to be determined. In this work, we focus on the progress determination of the latter, whereas issues to progress determination of single lifecycles processes have already been described in [4].

15th Central European Workshop on Services and their Composition, February 16–17, 2023, Hannover, Germany

*Corresponding author.

lisa.arnold@uni-ulm.de (L. Arnold); marius.breitmayer@uni-ulm.de (M. Breitmayer); manfred.reichert@uni-ulm.de (M. Reichert)

0000-0002-2358-2571 (L. Arnold); 0000-0003-1572-4573 (M. Breitmayer); 0000-0003-2536-4153 (M. Reichert)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

The remainder of this paper is structured as follows: Section 2 gives an overview of object-centric business processes and defines the overall goal of our research on monitoring such processes. Section 3 then defines fundamental research questions to refine the problem space. Section 4 gives a synopsis of related work. Finally, Section 5 closes this paper with a summary and outlook.

2. Background

An object-centric business process consists of multiple, interacting objects, whose behaviour is defined in terms of lifecycle processes (lifecycles for short). Each lifecycle has one start state and at least one end state as well as a number of intermediate states (cf. Figure 3). Each state can be refined by several steps that refer to update of object attributes [5]. Consequently, object-centric business process is data-driven. During run time, form sheets are generated automatically from the states of an object. Logically, steps represent the input fields of a form sheet.

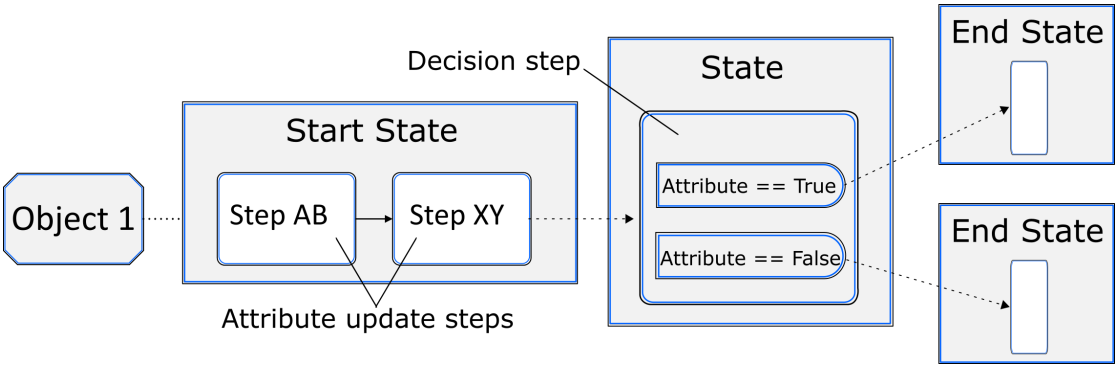


Figure 1: Structure of a simple lifecycle.

In turn, a relational process structure (RPS) (cf. Figure 2) represents the order and constraints for processing the objects and lifecycles, respectively, of the objects as well as the cardinalities between them [6]. Finally, coordination processes coordinate the execution business process (i.e., the sequence lifecycle states).

In [7], we have defined four research questions in the context of determining the progress of an object-centric business process. The aim of Research Questions 1 and 2 (RQ) was to determine the progress of a lifecycle in both a state-based view (i.e., by utilising the abstraction enabled by states) and in a step-based view, which refine the state-based view. These question were discussed in [4]. The Research Question 3 deals with determining the overall progress based on the results of Research Question 1 and 2.

RQ 1 How can the progress of a single lifecycle process with its state-based view form be determined?

RQ 2 How can the progress of the processing of a single state within a lifecycle process be measured?

RQ 3 How can the progress of multiple, interacting (i.e., interrelated) lifecycles be determined?

RQ 4 How does a coordination process affect the progress of an object-centric business process?

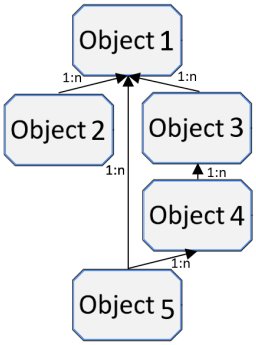


Figure 2
Simple RPS.

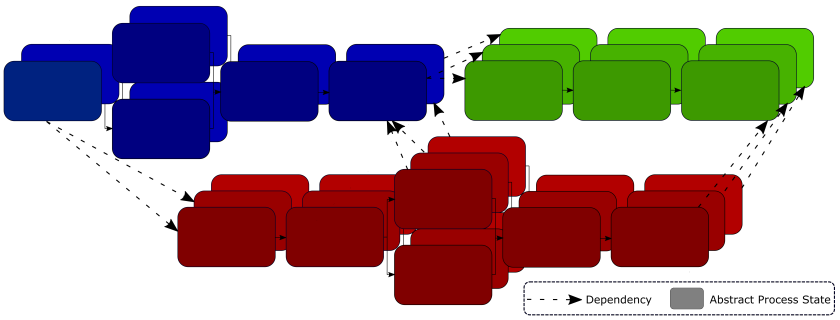


Figure 3
Simple big picture of running lifecycle instances.

Finally, Research Question 4 considered how coordination progress affects the progress determination of Research Question 3 [8].

3. Research Questions

Due to the high dynamics of relation process structures (e.g., due to varying numbers of interrelated objects or dynamic changes) Research Question 3 is challenging to address. For example, each deletion and addition of an object instance affects the progress of the respective process structures, i.e., the overall business process. In a nutshell, to determine the progress of multiple, interacting lifecycles the following sub-research questions need to be answered:

Sub-RQ 1 Does a generally suitable accepted understanding of progress exists?

Sub-RQ 2 How can the progress of a single object with multiple object instances be determined?

Sub-RQ 3 Which patterns are characteristic for an RPS and how can the progress of these patterns be determined?

Sub-RQ 4 How can the overall progress of an RPS be determined and how can this be accomplished in a way that matches the human intuition best?

Sub-RQ 5 How should the progress of an RPS be visualised to users such that the needs of individual user groups are met?

Sub-RQ 1 is mandatory and therefore the most important one of the given questions. If there exist no generally suitable accepted understanding of progress, the following four sub-research

questions can not be directly answered in a precise way. Therefore, we investigated Sub-RQ 1 in an empirical study with about 200 participants [9]. As key observation of this study, the majority of the participants gave very similar answers. For example, progress jumps within a displayed progress bar were rejected by most participants even though the overall progress of an RPS might decrease when adding dynamic objects and object relations (i.e., constraints). From [9] it may be concluded that a general understanding of progress exists. As this central question has been answered positively, we can focus on the issues addressed by Sub-RQ 2 to Sub-RQ 5. **Sub-RQ 2** deals with the progress of a single object and how this progress can be determined at run time taking dynamic changes into account as well. The latter include the unplanned creation of object instances (add objects) as well as their deletion during run time. **Sub-RQ 3** first identifies common patterns of an RPS and then progress determination for these specific patterns. **Sub-RQ 4** investigates how the overall progress of the RPS can be determined to match the human intuition best based on the results from the previous sub-research questions. Finally, **Sub-RQ 5** investigates the different possibilities of visualising the progress of multiple, interacting object instances for individual user groups.

4. Related Work

To the best of our knowledge there exist no works dealing with the progress of object-centric business process and its monitoring. In [10], an approach to measure the progress of activity-centric business process is presented. In [11], progress determination is based on data state transitions in activity-centric business process, improved by the use of object state transition [12]. Also, related to our work is predictive process monitoring, e.g. consider the detailed systematic literature review is presented in [13] to explore the current state of predictive process monitoring.

5. Conclusions

This paper discusses the challenges of determining the progress of RPSs, i.e., large, complex, and abstract process not being comprehensive to humans. To handle these challenges Research Question 3 "*How can the progress of multiple, interacting (i.e., interrelated) lifecycles be determined*?" (from [7]) needs to be refined. For this purpose, five sub-research questions were derived in this paper. By refining Research Question 3 we need to discuss this complex process structure in an early state.

References

[1] K. Andrews, S. Steinau, M. Reichert, Enabling runtime flexibility in data-centric and data-driven process execution engines, Information Systems (ISJ'21) (2021).

[2] K. Andrews, S. Steinau, M. Reichert, Enabling ad-hoc changes to object-aware processes, in: IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC'18), IEEE, 2018, pp. 85–94.

[3] K. Andrews, S. Steinau, M. Reichert, A tool for supporting ad-hoc changes to object-aware processes, in: 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW'18), IEEE, 2018, pp. 220–223.
[4] L. Arnold, M. Breitmayer, M. Reichert, A one-dimensional kalman filter for real-time progress prediction in object lifecycle processes, in: IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW'21), 2021, pp. 176–185.
[5] 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.
[6] 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.
[7] L. Arnold, M. Breitmayer, M. Reichert, Towards real-time progress determination of object-aware business processes, in: Proceedings of the 13th European Workshop on Services and their Composition (ZEUS'21), volume 2839, 2021, pp. 14–18.
[8] S. Steinau, V. Künzle, K. Andrews, M. Reichert, Coordinating business processes using semantic relationships, in: Conf. on Business Informatics (CBI'17), IEEE, 2017, pp. 33–42.
[9] L. Arnold, M. Breitmayer, M. Reichert, Progress determination of a bpm tool with ad-hoc changes: An empirical study, in: RCIS'22, Springer, 2022, pp. 107–123.
[10] A. Koschmider, J. L. d. l. Vara, J. Sánchez, Measuring the progress of reference model-based business process modeling, in: INFORMATIK 2010, 2010, pp. 218–229.
[11] N. Herzberg, A. Meyer, Improving process monitoring and progress prediction with data state transition events, Central European Workshop on Services and their Composition (ZEUS'13) (2013).
[12] N. Herzberg, A. Meyer, M. Weske, Improving business process intelligence by observing object state transitions, Data & Knowledge Engineering (DKE'15) (2015) 144–164.
[13] C. Di Francescomarino, C. Ghidini, F. M. Maggi, F. Milani, Predictive process monitoring methods: Which one suits me best?, in: Int. Conf. on Business Process Management (BPM'18), 2018, pp. 462–479.

Toward Model-driven Planning Support for Construction Processes

Anjo Seidel[1]

[1]Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert Str. 2–3, Potsdam, 14482, Germany

Abstract

Construction sites are shaped by knowledge-intensive, multi-instance, and item-dependent processes. The knowledge workers in such domains need to model the construction processes and construction elements and plan the future process execution collaboratively with all involved parties. However, it is challenging to model these processes comprehensively and accessibly for domain experts, while the planning of the future execution can not be fully supported yet. This position paper proposes the means toward developing suitable process modeling approaches, providing the opportunity to model process goals, and combining both to derive useful execution plans from analyzing the process model's state space in regard to the goal model.

Keywords

Knowledge-intensive Processes, Construction Processes, Process Modeling, Goal Modeling, Planning Support

1. Introduction

In recent years, construction sites are becoming more and more digitized; from mobile devices to 4D-Models of construction sites, and drones for supervising the construction progress [1].

The business process management community also started to investigate the use case of construction sites [2, 3]. In general, processes on construction sites can be considered knowledge-intensive [2] as they have the following characteristics[4], which are imposed by the volatile execution context of the processes. Knowledge-intensive processes are executed by knowledge workers such as construction workers and construction managers. Furthermore, knowledge-intensive processes are unpredictable and unrepeatable. They emerge as knowledge workers make decisions based on their domain knowledge. Planning is an important part of knowledge work. It means aligning the future execution of tasks toward a process goal. As weather conditions change, faults occur, and delays postpone future progress, plans have to be adapted and the knowledge workers need to re-plan.

More challenging, processes on construction sites can be considered multi-instance processes and item-dependent [3]. A house has multiple process instances running for each of its flats, which in turn influence each other's process execution. Also, all instances are item-dependent as the performed activities like painting jobs depend on existent material such as the paint and

15th Central European Workshop on Services and their Composition
anjo.seidel@student.hpi.de (A. Seidel)
0000-0002-9652-5340 (A. Seidel)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

a plastered wall.

Traditional modeling techniques like BPMN are not well suited to depict the complexity of such domains. Even the existing modeling approaches for construction processes [2, 3, 5] do not fulfill certain requirements of knowledge-intensive processes fully. For instance, late goal modeling and planning are an important part of knowledge work [4]. Planning is the alignment of actions according to a goal. With changing goals and execution contexts, execution plans on construction sites need to be re-planned frequently. If construction workers become sick, the delivery of the paint postpones, or simply the weather permits painting the house's facade, the painting activities have to be rescheduled. This might influence the overall progress and the schedule of the whole construction site needs to be replanned.

To tackle the problem of providing planning support for complex domains such as construction sites, we provide a research agenda that combines the analysis of the use case and the applicability of different process modeling approaches to it. We also need to allow the modeling and re-modeling of goals during design- and run-time. With the process model and the goal models at hand, the behavior of the process model can be analyzed in regard to the goals. The results can be used to generate execution plans for the future execution of the current case.

In the following Section 2, we outline the existing work related to our research endeavor. Section 3 discusses our approach and the planned contribution before Section 4 concludes the paper.

2. Related Work

Work related to our research agenda includes knowledge-intensive processes, modeling approaches toward construction processes, the modeling of knowledge-intensive processes, and planning support.

Knowledge workers face tasks of identifying and solving problems with a low level of standardization in knowledge-intensive processes [6]. Their execution context is vast and fast-changing [7]. Di Ciccio et al. provide an overview of the characteristics of knowledge-intensive processes and the requirements to support them [4]. However, it is challenging to model them comprehensively and they are often under-specified [7]. Additionally, to being knowledge-intensive, construction processes can be considered multi-instance and item-dependent [3] making modeling even more challenging.

Different approaches aim to model construction processes. First, van der Aalst et al. introduced case handling as a means to support the processes of a dutch construction company [2]. By now, new case management approaches have emerged and might also be applicable. Marengo et al. proposed the construction process modeling language CoPModL [3]. It combines the declarative modeling of activities and items on which these activities can be executed.

In the context of Lean Construction Management, different Tools have been developed to model and monitor the processes on construction sites [8].

Different modeling approaches aim to depict knowledge-intensive processes and might, therefore, be suitable to model construction processes. Hybrid modeling techniques combining the concepts of declarative and data-centric process modeling approaches are promising for the construction domain. Contemporary approaches are OCBC [9], RESEDA [10], and

ReGraDa [11]. Other approaches aim to implement case management [7] to support knowledge-intensive processes. Case Management and Model and Notation [12] and fragment-based Case Management [13, 14] were proposed.

Though late goal modeling is crucial for knowledge-intensive processes, it is only limitedly supported in most process modeling approaches [4]. In previous work, the means toward the modeling of goals for knowledge-intensive processes were provided [15, 16].

Another important aspect of knowledge work is planning. It is the alignment of actions to reach a goal. Different approaches aim at providing planning support. SmartPM [17] can be used for exception handling. Sprovieri and Vogler [18] compose process models from partly structured models. Other approaches by Wynn et al. [19] and Rozinat et al. [20] allow to simulate process models to provide decision support. Different predictive process monitoring approaches predict the next actions in running instances based on machine learning [21, 22]. Still, the contemporary approaches do not fully map the characteristics and requirements for knowledge-intensive processes. The first steps toward model-driven planning support by providing recommendations for the next best actions according to a goal were made in previous contributions [23, 16].

3. Planned Contribution

We endeavor to provide suitable planning support for knowledge-intensive, multi-instance, and item-dependent processes like construction processes. To pursue this goal, the general idea is to combine the process model with a goal model for the current process execution. The model's state space can then be analyzed to find possible execution sequences that satisfy the goals of the knowledge workers [16]. Analyzing goal-satisfying paths may result in knowledge about what next actions are suitable to reach the goal. This knowledge can be used to provide decision support [23]. In the future, comprehensive execution plans could be generated as well.

First, a suitable behavioral modeling technique to define the processes on construction sites is needed. Second, goals need to be modeled in that context. The state space analysis with both models needs to provide meaningful planning support in form of execution plans. This results in our following research questions.

RQ1 What is a suitable modeling language to depict construction processes comprehensively?

RQ2 How can goals be modeled in the context of construction processes?

RQ3 How can execution plans for construction sites be automatically generated from models and goals?

To answer **RQ1**, we plan to analyze the requirements for modeling knowledge-intensive, multi-variant, and item-dependent processes. We will investigate and compare contemporary approaches for modeling construction processes and knowledge-intensive processes. As a result, existing approaches can then be extended or new approaches can be developed to map these requirements. In our first informal analysis, case management [7], especially fragment-based case management [24] seemed promising. The concepts allow a notion of multi-instance processes while being able to depict the depending items. Yet, different adaptations have to be made to utilize these approaches fully.

Also part of the prior analysis is identifying the means to model goals, answering **RQ2**. Techniques to model goals can then be combined with the result of RQ1. A general approach toward the modeling of goals for knowledge-intensive processes was already provided [15, 16]. A constraints the state of the involved data and available actions in future execution states. However, the involved knowledge workers profit from more domain specific means to model goals. Such approaches could include the utilization of buiding information models (BIM), which provide 3D-models of buildings to be built. They are already well established in the domain and can depict the construction items in different execution states.

To address **RQ3**, the state space of the construction process models can be analyzed. Several challenges have to be overcome. State spaces of process models grow exponentially and can become infinitely large. However, the large state spaces also yields the possibility to find many suitable process executions that can be utilized to derive execution plans. However, it should be reduced as much as possible and suitable algorithms need to be developed that cope with their size and return suitable plans. The retrieved suitable execution plans can then be evaluated how resilient they are to changing execution context. In that context, machine learning techniques can be utilized to incorporate domain knowledge into the plans. These plans then need to be displayed comprehensibly. A usual way to display plans on construction sites are schedules with Gantt-charts. The resulting planning recommendations should balance the complexity of all possible execution sequences and comprehensibility.

All three research questions long for tooling support. We aim to develop prototypical implementations for construction process modeling, construction goal modeling, and the computation of planning recommendations out of the previous two. The suitability of the presented approach should be evaluated in the field by conducting case studies.

4. Conclusion

Construction processes are knowledge-intensive, multi-instance, and item-dependent. Planning is an important task of the involved knowledge workers, but only limitedly supported. This position paper proposes an approach to provide planning support for processes on construction sites and similar domains. The approach starts by analyzing the requirements for comprehensive construction process models. Such a modeling approach can then be developed and combined with suitable goal modeling for construction sites. Both process model and goal model can then be utilized to analyze the model's possible behavior in order to find execution plans that satisfy the goal. The result of this endeavor is model-driven planning support for construction sites and similar domains.

References

[1] F. Elghaish, S. Matarneh, S. Talebi, M. Kagioglou, M. R. Hosseini, S. Abrishami, Toward digitalization in the construction industry with immersive and drones technologies: a critical literature review, Smart and Sustainable Built Environment (2020).

[2] W. M. van der Aalst, M. Stoffele, J. Wamelink, Case handling in construction, Automation in Construction 12 (2003) 303–320.

[3] E. Marengo, W. Nutt, M. Perktold, Copmodl: Construction process modeling language and satisfiability checking, Inf. Syst. 103 (2022) 101457. doi:10.1016/j.is.2019.101457.

[4] C. D. Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches, J. Data Semant. 4 (2015) 29–57. doi:10.1007/s13740-014-0038-4.

[5] E. Marengo, W. Nutt, M. Perktold, Construction process modeling: Representing activities, items and their interplay, in: M. Weske, M. Montali, I. Weber, J. vom Brocke (Eds.), Business Process Management - 16th International Conference, BPM 2018„ volume 11080 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 48–65. doi:10.1007/978-3-319-98648-7_4.

[6] P. Pyöriä, The concept of knowledge work revisited, J. Knowl. Manag. 9 (2005) 116–127. doi:10.1108/13673270510602818.

[7] K. D. Swenson, Position: BPMN is incompatible with ACM, in: M. L. Rosa, P. Soffer (Eds.), Business Process Management Workshops - BPM 2012 International Workshops, volume 132 of *Lecture Notes in Business Information Processing*, Springer, Berlin, Heidelberg, 2012, pp. 55–58. doi:10.1007/978-3-642-36285-9_7.

[8] S. Singh, K. Kumar, Review of literature of lean construction and lean tools using systematic literature review technique (2008–2018), Ain Shams Engineering Journal 11 (2020) 465–471.

[9] W. M. P. van der Aalst, A. Artale, M. Montali, S. Tritini, Object-centric behavioral constraints: Integrating data and declarative process modelling, in: A. Artale, B. Glimm, R. Kontchakov (Eds.), Proceedings of the 30th International Workshop on Description Logics, volume 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1879/paper51.pdf.

[10] J. C. Seco, S. Debois, T. T. Hildebrandt, T. Slaats, RESEDA: declaring live event-driven computations as reactive semi-structured data, in: 22nd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2018, IEEE Computer Society, 2018. doi:10.1109/EDOC.2018.00020.

[11] L. Galrinho, J. C. Seco, S. Debois, T. T. Hildebrandt, H. Norman, T. Slaats, Regrada: Reactive graph data, in: F. Damiani, O. Dardha (Eds.), Coordination Models and Languages - 23rd IFIP WG 6.1 International Conference, volume 12717 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 188–205. doi:10.1007/978-3-030-78142-2_12.

[12] M. Kurz, W. Schmidt, A. Fleischmann, M. Lederer, Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management, in: J. Ehlers, B. Thalheim (Eds.), Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, S-BPM ONE 2015, ACM, 2015, pp. 4:1–4:9. doi:10.1145/2723839.2723843.

[13] M. Hewelt, M. Weske, A hybrid approach for flexible case modeling and execution, in: M. L. Rosa, P. Loos, O. Pastor (Eds.), Business Process Management Forum - BPM Forum 2016, volume 260 of *Lecture Notes in Business Information Processing*, Springer, Cham, 2016. doi:10.1007/978-3-319-45468-9_3.

[14] S. Haarmann, M. Montali, M. Weske, Refining case models using cardinality constraints, in: M. L. Rosa, S. W. Sadiq, E. Teniente (Eds.), Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, volume 12751 of *Lecture Notes in Computer Science*, Springer, Cham, 2021, pp. 296–310. doi:10.1007/978-3-030-79382-1_18.

[15] S. Haarmann, A. Seidel, M. Weske, Modeling objectives of knowledge workers, in: A. Marrella, B. Weber (Eds.), Business Process Management Workshops - BPM 2021 International Workshops, Revised Selected Papers, volume 436 of *Lecture Notes in Business Information Processing*, Springer, 2021, pp. 337–348. URL: https://doi.org/10.1007/978-3-030-94343-1_26. doi:10.1007/978-3-030-94343-1_26.

[16] A. Seidel, S. Haarmann, M. Weske, Model-based decision support for knowledge-intensive processes, Journal of Intelligent Information Systems (2022) 1–23.

[17] A. Marrella, M. Mecella, S. Sardiña, P. Tucceri, SmartPM: Automated adaptation of dynamic processes, in: F. Toumani, B. Pernici, D. Grigori, D. Benslimane, J. Mendling, N. B. Hadj-Alouane, M. B. Blake, O. Perrin, I. Saleh, S. Bhiri (Eds.), Service-Oriented Computing - ICSOC 2014 Workshops - WESOA; SeMaPS, RMSOC, KASA, ISC, FOR-MOVES, CCSA and Satellite Events, Paris, France, November 3-6, 2014, Revised Selected Papers, volume 8954 of *Lecture Notes in Computer Science*, Springer, Cham, 2014, pp. 423–427. doi:10.1007/978-3-319-22885-3_40.

[18] D. Sprovieri, S. Vogler, Run-time composition of partly structured business processes using heuristic planning, in: International Conference on Enterprise Systems, ES 2015, Basel, Switzerland, October 14-15, 2015, IEEE, New York, 2015, pp. 225–232. doi:10.1109/ES.2015.30.

[19] M. T. Wynn, M. Dumas, C. J. Fidge, A. H. M. ter Hofstede, W. M. P. van der Aalst, Business process simulation for operational decision support, in: A. H. M. ter Hofstede, B. Benatallah, H. Paik (Eds.), Business Process Management Workshops, BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers, volume 4928 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2007, pp. 66–77. doi:10.1007/978-3-540-78238-4_8.

[20] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, C. J. Fidge, Workflow simulation for operational decision support, Data Knowl. Eng. 68 (2009) 834–850. doi:10.1016/j.datak.2009.02.014.

[21] I. Teinemaa, M. Dumas, M. L. Rosa, F. M. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Trans. Knowl. Discov. Data 13 (2019) 17:1–17:57. doi:10.1145/3301300.

[22] C. D. Francescomarino, C. Ghidini, F. M. Maggi, F. Milani, Predictive process monitoring methods: Which one suits me best?, in: M. Weske, M. Montali, I. Weber, J. vom Brocke (Eds.), Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, volume 11080 of *Lecture Notes in Computer Science*, Springer, Cham, 2018, pp. 462–479. doi:10.1007/978-3-319-98648-7_27.

[23] A. Seidel, S. Haarmann, Decision support for knowledge-intensive processes, 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. 20–29. URL: http://ceur-ws.org/Vol-3113/paper5.pdf.

[24] S. Haarmann, WICKR: A Joint Semantics for Flexible Processes and Data, Ph.D. thesis, Universität Potsdam, 2022.

Improving Load Balancing of Long-lived Streaming RPCs for gRPC-enabled Inter-service Communication

Christopher Starck[1,†], Javad Ghofrani[1,*,†]

[1]Institute of computer engineering, University of Luebeck, Luebeck, Germany

Abstract

Many companies adopted gRPC since 2015 for their microservices architecture due to its efficient communication method, particularly regarding streaming data. However, streaming requires long-lived connections that load-balancing solutions fail to address. These problems include unexpectedly overloading services, which may result in a chain of server failures. This paper describes the load-balancing problems with long-lived streaming RPCs and proposes a flow control-inspired approach that shifts the control of incoming requests from the load balancers to the application servers. Results of our experiments show that utilizing this method leads to more server resiliency in the case of long-lived streaming RPCs.

Keywords

microservice architecture, inter-service communication, gRPC-streaming, load balancing

1. Introduction

gRPC is an open-source Remote Procedure Call (RPC) framework introduced by Google in 2015 that provides a high-performance alternative to REST [1]. Companies reported improved developer productivity with gRPC, as it allows for easier integration with existing legacy systems[1] and provides automatic code generation for various programming languages [2]. It utilizes HTTP/2 [3] for binary communication, which is more efficient than the text-based protocol used by REST. The built-in flow control and error-handling features make gRPC more reliable and a viable choice for inter-service communication in microservice architectures [4].

gRPC provides streaming RPCs between servers and clients, enabling them to keep the connection and continue with bidirectional communication once they opened it. This functionality makes the current load balancing mechanisms [5] insufficient since the load balancer will be involved only in the connection establishment step. After starting the connection, the load balancer will have no control over requests and loads on the server. This way, the clients can suddenly increase the load on the server and cause the server to fail.

15th Central European Workshop on Services and their Composition

*Corresponding author.

†These authors contributed equally.

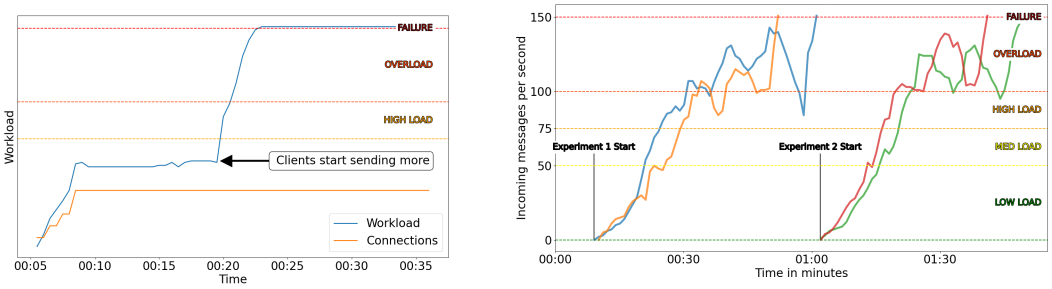
christopher.starck@googlemail.com (C. Starck); javad.ghofrani@gmail.com (J. Ghofrani)

https://iti.uni-luebeck.de/ (J. Ghofrani)

0000-0001-7298-2080 (C. Starck); 0000-0002-9249-7434 (J. Ghofrani)

[1]https://www.cncf.io/case-studies/netflix/

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org



(a) Sudden workload increase due to streaming RPCs.

(b) Servers when one of them crashes

Figure 1: Example of Sudden workload change in clients which leads to cascading failure of the servers

This paper highlights the load-balancing challenge of streaming RPCs in gRPC. We devise a flow control-inspired solution to handle this challenge. Handling such cases is important since crashing the streaming servers can lead to cascading server failure, loss of sensitive data, and decreased service quality and customer satisfaction.

This paper is structured as follows: We explain the problem of long-lived streaming RPCs with the load balancing mechanism in Section 2. In Section 3, we review the existing literature on gRPC. In Section 4, we propose our approach inspired by flow control mechanisms, and finally, we summarize our findings and recommendations for future research in Section 5.

2. Load Balancing Problems with Long-lived Streaming RPCs

There is no clear definition for when a streaming RPC can be considered long-lived. Depending on the use case, long-lived RPCs may last a few seconds in real-time scenarios or for several days in business scenarios. Streaming RPCs involve the exchange of zero or more messages over a single connection. TCP enables these types of RPCs through full-duplex mode, where messages can be sent simultaneously in both directions. Whereas client and server streaming are half-duplex, i.e. only one entity can send messages at a time. Additionally, connections are only half-closed, which allows requests to be sent over the same connections multiple subsequent times [2]. This kind of reusing of connections impairs the load balancing mechanisms which dispatch the client request to the servers.

Once a client sends a streaming RPC request to a server, the load balancing mechanism checks the server's health status and dispatches the request to the server with less working load. After accepting a streaming RPC request, the client can keep the connection open and reuse it anytime. The server overloads and crashes if many clients resume their open-held connections and start streaming. This way, the load balancing mechanism will be bypassed and cannot handle the load on the server. This can happen if a sudden increase in a server's workload causes that server to fail. All clients connected to that server will receive an error and try to send a request to another available server. The servers that end up receiving incoming requests have an increased workload. In the worst case, these servers may also fail because of the sudden increase in workload. This server could also find itself overloading due to an

unexpected increase in connections. This particular problem can be categorized as a chain of server failures [6]. Figure 1 illustrates this scenario.

3. Related Work

While gRPC is a well-used technology in the industry, to the best of our knowledge, existing research in this area is limited to a few research studies on Lee and Liu [7], Chamas et al. [8], and Shah et al. [9]. Indrasiri and Kuruppu [2] also published a book on gRPC that introduces building cloud-native applications with Go and Java and discusses gRPC as inter-service communication technology.

Lee and Liu [7] present a general approach for migrating APIs from REST to gRPC. Their motivation is to improve communication performance by leveraging the features of gRPC. They developed a manual refactoring workflow with six steps and identified two directions for improvement. The first direction is automation, which could streamline the migration process. The second direction is the inclusion of error handling, authentication, and load balancing, which could improve the robustness of the migrated API. While their approach provides a helpful starting point for organizations looking to migrate to gRPC, it does not address the challenge of load-balancing long-lived streaming RPCs that we identified in our work.

Chamas et al. [8] study the energy consumption of various sorting algorithms with varying input sizes and types in the context of computation offloading mobile applications. They compare four communication protocols for remote execution: SOAP, REST, sockets, and gRPC. They reported that the local execution is generally more economical for small inputs, with a few exceptions for object input types. Their study provides valuable insights into the performance of different communication protocols in computation offloading, including the poor performance of gRPC.

Shah et al. [9] provide an overview of load-balancing algorithms in cloud computing. They noted that the load on servers increases as more applications move to or run on the cloud. Increasing the load leads to the problem of over-utilized and under-utilized servers, which necessitates load balancing. They include several static and dynamic load-balancing algorithms to address the resource allocation problem in data centers.

4. Flow control Instead of Load Balancing

Our approach is based on the idea of flow control. We utilize the sliding window method to manage a server's workload dynamically. For this purpose, the server holds a simplified internal model of the workload where the sliding window counts ongoing requests over the last t_w seconds. We use this information in our experiments to discretize the workload into LOW LOAD, MEDIUM LOAD, HIGH LOAD, OVERLOAD and SYSTEM FAILURE.

We implemented our approach using bidirectional streaming RPCs exclusively. This mode of communication allows the server to request messages from the client. We impose a constraint on the client where it must wait before sending a message after the initial message. We achieve this by introducing a special field to each message from server to client. This field holds an integer value that specifies the number of messages the client is allowed to send. Whenever

a client receives a response message with a non-zero value, it can send another message or complete the call. If the field is zero, the client does not send a message and continues to wait.

These semantics allow the server to suspend and resume individual connections to prevent itself from overloading. The server suspends connections in an overloading state and resumes suspended connections otherwise.

4.1. Experiments

In our experiments, a server shuts itself down if its internal workload model is SYSTEM FAILURE. This way, we can simulate real-world conditions via small-scale experiments. We performed our experiment on Ubuntu 22.04 (64 bit) running on a system equipped with Intel(R) Xeon(R) Gold 6254 CPU (3.10GHz), 8 Cores, and 16 GB of RAM. Furthermore, we used Docker Engine Version (v20.10.21) with *buildkit* enabled and Docker Compose Version (v2.12.2).

We utilize docker swarm networking with a round-robin load balancing policy for service discovery. We intentionally use a basic load balancing setup to illustrate how our approach performs. With this setup, we can imperfectly split the connections between our servers. This imperfection is a helpful approximation for real-world scenarios. The clients were configured to send a specific amount of messages per second. Finally, clients implement a simple retry policy which repeats failed requests up to 3 times before giving up.

$$\text{Messages per Second} = \frac{\text{Number of Clients}}{\text{Number of Servers}} \times \frac{\text{Number of Messages}}{\text{Duration of Requests}} \tag{1}$$

We run several experiments with an increasing number of messages. The experiments are designed to push the workload beyond the failure threshold. First, we test that our simulation functions correctly with fewer messages. Second, we increase the number of messages to the limit of what our servers should be able to handle. This tests the ability of our approach to handle periods of increased workloads. Finally, we increase the number of messages well above the intended limits to see how our approach scales. By controlling the number of messages, we approximate a real-world scenario of thousands of clients within our resource-constrained testing hardware. We published our experiment code in a public GitHub repository [10] and included steps to reproduce it.

Parameter Name	Configuration
Number of Servers	2
Number of Clients	50
Number of Requests	5
Number of Messages	20,40,60,80,100
Duration of Requests	20 seconds
Failure Threshold	150 Messages per second

Table 1
Configuration for the experiments.

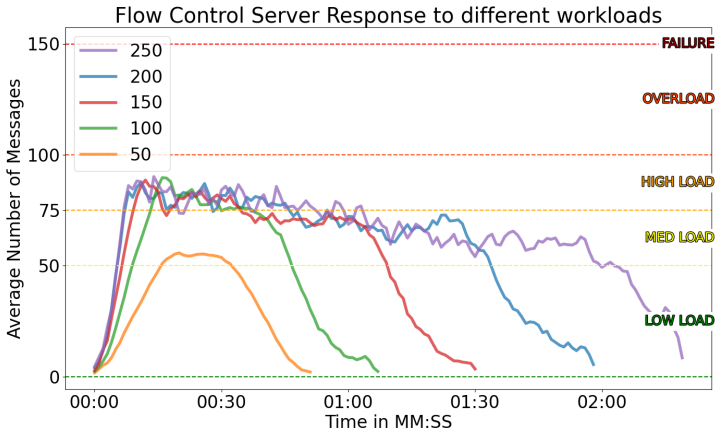


Figure 2: Flow Control server Response for various workloads. The lines represent different levels of expected workloads. Here, workload refers to incoming messages per second.

4.2. Results

Workloads above their threshold for system failure are handled robustly, far from an overloading state. The flow control mechanism can explain the server's response. When the server starts to overload, it momentarily suspends incoming requests. These requests still count towards the threshold, but subsequent requests are prevented until the workload subsides. Figure 2 shows resilient behavior to the increased workloads. The average number of messages per request decreases as the first clients complete their requests. This begins to happen when the global workload decreases. Experiments with higher initial workloads show a steeper decline in the average number of messages towards the end because each client sends a larger volume of messages.

No failed requests suggest that shifting the control from the load balancer to the application server is an effective way to handle sudden increases in incoming traffic.

We notice, however, that the requests take longer to be processed under high load, which is to be expected. In Table 2, we can see that the Round Trip Time (RTT) increases for higher workloads. The trade-off for a reliable but slower application may be worthwhile in environments where these use cases are essential.

Workload	Average RTT
50	9.52ms
100	11.98ms
150	18.03ms
200	24.77ms
250	32.88ms

Table 2
Average Round Trip Time (RTT) for different workloads. Each experiment was repeated at least 40 times.

4.3. Discussion

In Figure 2, we can see the normal response for the lowest workload in orange and the flow-controlled response for all other workloads. We find that the flow-controlled response behaves differently in preventing the sudden increase in workload. Additionally, we note that, on average, it stays below the overload threshold because the server suspends connections successfully, which further limits the workload.

The number of messages is an incomplete model of a server's workload. It is inaccurate, but it does suffice in testing our load balancing mechanism on a small scale.

We showed that an incomplete model on a resource-constrained single host machine handles workloads beyond its configured limits. Our experiment setup uses a basic configuration of the popular orchestration framework docker compose.

The ideas of our proposed solution can be generally applied to any gRPC application because it only requires a change in protobuf service definitions. Furthermore, it does not presume any specific load balancing setups and even works between a server and a client without any load balancing entity.

5. Conclusion

In this paper, we raised the issues of load balancing in long-lived streaming RPCs and explained the inability of conventional load balancing mechanisms to handle it. To solve this issue, we proposed an approach based on shifting the control over incoming requests from load balancer to the application server. This method prevents the server from overloading, a desirable feature for inter-service communication in microservice-based architecture. The management of incoming requests proves to be an effective measure in preventing server overload and cascading server failure, a chain of events where servers failing lead to others failing due to a sudden increase in re-transmitted messages from clients.

The workload model could include more parameters to better model real-world scenarios. With an extended workload model, more sophisticated flow control mechanisms should be explored. A non-linear controller could lead to further improvements where the server would try to stay close to a desired state. The performance impact on the overhead the server incurs remains to be studied.

Applying L7 Flow Control to a real-world application could provide valuable information about its effects on availability and scalability. In real-world scenarios, there are often many services and many different types of RPCs in use. Modifying the protocol buffers for each service and RPC may not be a practical solution.

Another improvement is to develop a framework that is transparent to its users. However, this requires extensive development and expertise in multiple programming languages to work seamlessly across different frameworks in gRPC. There is the possibility of introducing flow control mechanisms on a lower layer. TCP natively supports flow control [11], and gRPC also supports it at the framework level.

Although our applied flow control mechanism shows promising results, future research is required. Future research could focus on finding other ways to implement the ideas of L7 Flow Control.

References

[1] R. T. Fielding, Architectural styles and the design of network-based software architectures, University of California, Irvine, 2000.

[2] K. Indrasiri, D. Kuruppu, gRPC: Up and Running - Building Cloud Native Applications with Go and Java for Docker and Kubernetes, "O'Reilly Media, Inc.", Sebastopol, 2020.

[3] D. Stenberg, Http2 explained, 2014.

[4] A. Bucchiarone, N. Dragoni, S. Dustdar, P. Lago, M. Mazzara, V. Rivera, A. Sadovykh, Microservices, Science and Engineering. Springer (2020).

[5] R. Kaur, P. Luthra, Load balancing in cloud computing, in: Proceedings of international conference on recent trends in information, telecommunication and computing, ITC, Citeseer, 2012.

[6] M. T. Nygard, Release It! Design and Deploy Production-Ready Software, Pragmatic Bookshelf, Raleigh, NC, 2007.

[7] Y. Lee, Y. Liu, Using refactoring to migrate rest applications to grpc, ACM SE '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 219–223. doi:10.1145/3476883.3520220.

[8] C. L. Chamas, D. Cordeiro, M. M. Eler, Comparing rest, soap, socket and grpc in computation offloading of mobile applications: An energy cost analysis, in: 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), 2017, pp. 1–6. doi:10.1109/LATINCOM.2017.8240185.

[9] J. Shah, K. Kotecha, S. Pandya, D. Choksi, N. Joshi, Load balancing in cloud computing: Methodological survey on different types of algorithm, in: 2017 International Conference on Trends in Electronics and Informatics (ICEI), 2017, pp. 100–107. doi:10.1109/ICOEI.2017.8300865.

[10] C. Starck, J. Ghofrani, Improving Load Balancing of Long-lived Streaming RPCs for gRPC-enabled Inter-service Communication, 2023. URL: https://github.com/BalticBytes/grpc-load-balancing-long-lived-streaming-rpcs. doi:10.5281/zenodo.7641639.

[11] J. F. Kurose, K. W. Ross, Computer networking: a top-down approach, 6th ed ed., Pearson, 2013.

Immutable Operating Systems: A Survey

Sebastian Böhm[1,*], Guido Wirtz[1]

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

Abstract

Immutable Operating Systems, also called Container Operating Systems, are a new term in the area of operating systems. This type of operating system claims to be immutable, reliable, resilient, and even more secure compared to traditional operating systems. The number of scientific publications regarding Immutable Operating Systems is still limited. There is no widely accepted conceptualization yet. Therefore, this work aims to close this research gap. We utilize a literature review on a subset of solutions to derive the concepts. We use the gained insights to provide a general conceptualization, an overview of use cases and limitations, and finally, a definition of the term. An Immutable Operating System is a special type of operating system, primarily a minimal Linux distribution that introduces read-only file systems, automatic atomic updates, rollbacks, declarative configuration, and workload isolation to achieve higher reliability, scalability, and security, especially to serve as a container host. We conclude that this particular type of operating system provides noteworthy enhancements. It can reduce the maintenance efforts in distributed and heterogeneous areas, like edge and fog computing.

Keywords

Immutable Operating System, Container Operating System, Operating System, Linux

1. Motivation

Immutable Operating Systems (IOSs), also called Container Operating Systems (COSs), are an emerging trend in the area of Operating Systems (OSs) research [1–5]. Over the years, several Linux distributions have been published, introducing new concepts to fulfill the claim of being described as an IOS or COS. Specifically, openSUSE MicroOS (oMOS) [1], Fedora CoreOS (FCOS) [6], Flatcar Container Linux (FCL) [3], AWS Bottlerocket OS (ABOS) [7], and Talos Linux (TL) [5] are used for server-centric applications. Whereas in traditional OSs, all files are potentially modifiable by the system or processes, IOSs have a read-only root file system. This leads to many open questions about how an OS can be maintained without the chance to apply changes, like performing updates and writing log files during operation. So far, there is a lack of peer-reviewed scientific literature to provide a common understanding, characterization, and application scenarios of IOSs. Therefore, this paper aims to provide insights into IOSs. This addresses a definition, the concepts, and the technologies used by those systems. In addition,

Hannover'23: 14th Central European Workshop on Services and their Composition (ZEUS 2023), February 16–17, 2023, Hannover, Germany

*Corresponding author.

sebastian.boehm@uni-bamberg.de (S. Böhm); guido.wirtz@uni-bamberg.de (G. Wirtz)

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

https://www.uni-bamberg.de/en/pi/team/wirtz/ (G. Wirtz)

0000-0003-3719-1923 (S. Böhm); 0000-0002-0438-8482 (G. Wirtz)

CEUR Workshop Proceedings (CEUR-WS.org)

S. Böhm and D. Lübke (Eds.): 15th ZEUS Workshop, ZEUS 2023, Hannover, Germany, 16–17 February 2023, published at http://ceur-ws.org

there is no comprehensive overview of typical use cases and limitations, where IOSs play their strengths. Therefore, this contribution addresses the following research questions:

RQ1: What concepts do Immutable Operating Systems follow?

RQ2: What are typical use cases for Immutable Operating Systems?

RQ3: How can an Immutable Operating System be defined?

We perform qualitative literature research to obtain the essential aspects. This procedure allows us to identify already available solutions and alternative identifiers for the area of IOSs. To answer **RQ1**, we analyze the documentation of the identified solutions to assemble a comprehensive list of used concepts and technologies. In addition, we can get insights about the supposed application scenarios of those systems (**RQ2**). Finally, referring to the previous results, we can provide a first definition of IOSs (**RQ3**).

This paper is structured as follows: Section 2 clarifies the term immutability. In Section 3, we take a look at related works. We present our review on IOSs and their features in Section 4. This allows deriving a conceptualization in Section 5. Finally, we conclude our contribution with the limitations of our study (Section 6) and the conclusion (Section 7).

2. Immutability

According to the Cambridge Dictionary [8], immutability means "the state of not changing, or being unable to be changed". In computer science, the term refers mainly to the area of Object-Oriented Programming (OOP). Programs that follow the OOP paradigm consist of objects with state and behavior. A set of attributes represents the state. Also, objects have a particular behavior by providing methods, which may allow the modification of the objects' state [9]. Immutable objects do not allow any modification after the object has been initialized. To derive a modified version of the object, a new (immutable) copy of the object must be created. Conveyed to an OS, the state must not change during runtime. This means that modifications may result in destroying and redeploying the entire system with the updated configuration [10].

3. Related Work

Since the term IOS is relatively new, we could find only one peer-reviewed contribution by [11]. It refers to the LinuxKit project [12] that discusses the fundamental potentials of Immutable Servers. The main motivation is the avoidance of configuration drifts, where multiple servers with the same image, configuration, and modifications vary over time nonetheless [13]. The work of [11] discussed fundamental requirements for building an immutable server that were also used to design LinuxKit. Although it mentioned a subset of characteristics of immutable servers that are helpful in getting a first basic understanding and characterization of IOSs, it did not perform a review and evaluation of alternative IOSs to derive these characteristics. Furthermore, the concepts of alternative IOSs were not regarded, which limits the possibility and validity of deriving a general conceptualization. The work mentioned only a few alternative IOSs. Therefore, this work aims to provide a more detailed investigation of IOSs by performing a more comprehensive literature review to close this gap.

4. Review: Immutable Linux Distributions

We started our literature review by using Google Scholar, IEEEXplore, and ACM Digital Library with the search terms `Immutable (Linux [Server] | Operating System | Infrastructure)`. This revealed a GitHub repository [14] that contains a collection of immutable Linux distributions. Based on this collection, we checked and added related Linux distributions, which fulfill the claim of IOSs. We excluded all Linux distributions focusing only on desktop environments because our work targets distributed environments (Section 1). We considered five Linux distributions that are explicitly claiming to be immutable for a server-centric application (Table 1).

Immutable (Root) File System. Most of the IOS provide a fully immutable root file system [1, 4, 5, 15]. An exemption is FCL, which provides a writeable root file system for all types of data, for example, container images. However, all system-related files are mapped to the immutable directory mounted at `/usr`. Most IOSs keep some directories, like `/etc`, `/var`, `/home`, and `/root` accessible for read-write operations [4, 15–17]. These directories are used to store configurations, log files, and user data [18]. Nevertheless, OS-related files are still immutable.

Atomic Updates and Rollbacks. Since the root file system is fully immutable, updates must be installed in a particular way. According to the definition of immutability (Section 2), the state can not be changed. Changes must be applied on a copy of the OS. The distributions achieve this in different ways: oMOS uses `transactional-update` [19]. Technically, a new snapshot of the OS is created by using the underlying copy on write file system btrfs [20]. Then the desired changes are applied, and it can be booted from the newly created snapshot. In case of issues during a reboot, the file system can be reverted to the last known working snapshot [19]. FCOS uses `rpm-ostree` [21] that is a hybrid image and package system. It manages the OS files in a git-like manner with a content-addressed object store and checksums. This allows managing the OS as a repository that enables atomic updates by committing changes and rollbacks by reverting changes [21]. FCL, ABOS, and TL perform atomic updates via an A-B schema. These distributions provide two partitions for the OS, one active partition for the currently running system, and one passive one for upgrades. Once an update process is started, an updated OS base image is downloaded to the passive partition that can then be taken on the next reboot. In case of a failed boot process, the previous working partition is used for the rollback [4, 19, 22, 23]. All IOSs follow the standard release model, where at least a stable and testing branch can be selected [4, 23–25]. In addition, oMOS focuses on a rolling release model [26].

Table 1
Features of Immutable Operating Systems

	openSUSE MicroOS	**Fedora CoreOS**	**Flatcar Container Linux**	**AWS Bottlerocket OS**	**Sidero Labs Talos Linux**
Initial Release	2018	2018	2018	2020	2018
Release Model	Rolling / Standard	Standard	Standard	Standard	Standard
Immutable (Root) File System	✓	✓	✗, only `/usr`	✓	✓
Atomic Updates / Rollbacks	`transactional-update`	`rpm-ostree`	A-B	A-B	A-B
Automated Updates / Rollbacks	✓ / ✓	✓ / ✗	✓ / ✓	✓ / ✓	✗ / ✓
Declarative Configuration	Ignition, cloud-init	Ignition	Ignition	API, Custom	API, Custom
Workload Isolation	Podman	Podman, Docker	Docker	Docker	Docker, Kubernetes

Automated Updates / Rollbacks. IOSs target an unattended operation [1]. Regular updates should be done automatically. Nearly all of the investigated solutions support automated updates, also based on a schedule [4, 25–27]. Only TL requires a manual update [23].

Since updates may be disruptive, rollbacks to the previous working state should be automatically performed for a seamlessly unattended operation. Mostly all Linux distributions check if the system is booting after an update with the help of integrated low-level tools [4, 22, 23, 26]. FCOS requires manual intervention if an update fails [27].

Declarative Configuration. IOSs should be usable in different environments, like bare-metal, virtualized, or cloud environments. This requires a flexible way of configuration to meet the requirements of the target environment (Section 1). For example, usernames, passwords, SSH public keys, and network information must be set during the first boot process. A declarative configuration is also necessary to easily realize unattended installations for mass deployments. All of the solutions require and provide different tools for declarative configuration. The most popular solution is the tool Ignition [28] that is supported by oMOS, FCOS, and FCL [16, 29, 30]. Configuration tools, in general, allow the modification of disks during early boot. This comprises modifying the disk partitioning, adding files, and basic system configurations as mentioned above [28]. oMOS allows further cloud-init [31] that is also quite popular for declarative configuration [16]. ABOS and TL provide vendor-specific solutions, which are applied during the boot process. Besides that, both can be configured via an API [4, 32].

Workload Isolation. All presented Linux distributions strongly encourage the users to use container technology. This way of virtualization offers workload isolation between different applications that contributes to managing security concerns [33]. Due to the high popularity of OS-based virtualization, a large number of container runtimes, engines, and orchestrators emerged over the recent years [34]. The distributions that are part of this work offer different high-level container engines or orchestrators. Common is Podman [35], for example used by oMOS and FCOS [26, 36]. Docker [37] is also frequently used, for example, by FCL and ABOS [4, 38]. TL itself is designed to run Kubernetes [39], which is integrated into the OS [5]. ABOS does also provide a version that is equipped with Kubernetes by default [4].

5. Immutable Operating Systems

After investigating the core features of IOSs, we can summarize them in general concepts (Section 5.1). Afterward, we highlight use cases and limitations (Section 5.2). Finally, we conclude this chapter by providing a detailed definition, based on the obtained insights (Section 5.3).

5.1. Concepts

Reliability. According to the Institute of Electrical and Electronics Engineers (IEEE), software reliability is defined as "the probability that software will not cause the failure of a system for a specified time under specified conditions" [40, p. 17]. The following two criteria successfully contribute to the reliability of IOSs in a broader sense:

Immutability. Immutability means that all OS-related files can not be modified during runtime (Section 2). That implies that a working version of the OS runs consistently on every boot [1, 4, 15, 17]. Furthermore, configuration drifts, as described in Section 3, can be effectively

avoided [5]. This contributes to reliability because fewer fundamental changes during runtime may avoid crashing system services and user workloads.

Resiliency. The term resiliency can be defined as "the capacity of a system [...] to remain reliable, [...] in case of any malicious or accidental malfunctions or failures that result in a temporal or permanent service disruption" [41, p. 1]. IOSs offer resilience by atomic updates that reduce the likelihood of making the system inoperable. Some Linux distributions automatically verify updates and perform a rollback to the previous working version if an update was not successful. This rollback usually occurs if the OS is no longer able to boot.

Scalability. Scalability is described as follows: "The concept connotes the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement" [42, p. 17]. In the context of IOSs, it is possible to use them as candidates for scaling out and provisioning a large-scale infrastructure. The following two characteristics contribute to scalability:

Minimality. IOSs provide minimal installation images for bare-metal, virtualized, or cloud environments. They only contain a minimal set of software binaries and libraries to allow for running containerized workloads [3, 5, 26, 43]. This also contributes to security-related aspects because the attack surface is notably reduced.

Configurability. An IOS should be run at scale as a container host. This requires a comfortable way of configuration at scale because mass deployments should be possible. A file-based declarative configuration is an essential aspect of IOSs. Furthermore, using an API to register configuration changes during runtime leads to higher configuration flexibility.

Security. All workloads are executed by container virtualization. As already pointed out in the former section, containerization offers security enhancements by isolating workloads from each other. However, there is still the chance for privilege escalation due to vulnerabilities in kernel, container runtimes, and container engines, which allow access to the whole system. Due to the immutable design, it is at least not possible to manipulate the OS, for example, by replacing system libraries. This is a considerable security enhancement. Furthermore, automated updates keep the systems always up-to-date, to avoid security vulnerabilities.

5.2. Use Cases and Limitations

IOSs are intended to operate as container hosts. All of the platforms explicitly state that they provide a minimal OS with preinstalled and preconfigured container runtimes and container engines. Hence, IOSs can be used as standalone container hosts or in clusters, as implemented by TL. IOSs may foster the complexity reduction in edge and fog computing, where many heterogeneous nodes need to be equipped with container virtualization capabilities [44]. Many of the cloud providers adopted the benefits of these systems by providing their solutions, like Google [45], AWS [4], and Microsoft [46].

IOSs limit the modifiability in an extensive way. Mostly, there is no package manager available, or the distributor recommends not installing packages. The usage of applications that can not be run containerized might be limited. The modification of the root file system is not possible during runtime. Only selected locations are allowed to be written [4, 15, 16], depending on the Linux distribution (Section 4). Consequently, the proposed solutions are not fully immutable in a technical sense. A fully IOS cannot be used because process and application data changes

over time [11]. The update of system-related services requires a reboot. For distributions that use an A-B schema for updates, more storage might be required, because the image of the new version must be downloaded and installed.

5.3. Definition

Based on the former qualitative evaluation that has been derived from the official documentation of the different OS vendors, a definition of the term IOS can be derived:

An Immutable Operating System is a special type of operating system, primarily a minimal Linux distribution that introduces read-only file systems, automatic atomic updates, rollbacks, declarative configuration, and workload isolation to achieve higher reliability, scalability, and security, especially to serve as a container host.

They are usually built on top of an immutable root file system, perform atomic updates and rollbacks mostly automatically, and are configured in a declarative way to allow for an unattended operation in distributed areas. They leverage advanced file systems, tools, or strategies to realize these mechanisms. Instead of installing workloads with packages, they require the users to provide containers. Whereas not fully immutable in a technical sense, because there are still areas that are allowed to be written, they offer considerable enhancements.

6. Threats to Validity

We conducted literature research with no comprehensive qualitative comparison between the different Linux distributions and no quantitative evaluation, like a performance comparison. To gain our insights, we considered only a subset of available solutions and second only a subset of features to conceptualize and define IOSs. Besides a limited set of peer-reviewed literature, we relied only on the vendors' documentation. Some of the Linux distributions do not have an official release status yet. Fundamental design decisions and used tools of these systems might change rapidly, and the results and findings of this work might expire.

7. Conclusion

This contribution provides a first conceptualization of IOSs (**RQ1**). They have a minimal and immutable root file system, can perform atomic updates and rollbacks, can be configured and modified in a declarative way, and use container virtualization for workload isolation. Typically, they operate as container hosts, expecting to have the primary workload in a containerized form available (**RQ2**). Finally, we can define an IOS as a special type of OS, primarily a minimal Linux distribution that introduces read-only file systems, automatic atomic updates, rollbacks, declarative configuration, and workload isolation to achieve higher reliability, scalability, and security, especially to serve as container host (**RQ3**).

We want to expand our work by evaluating further aspects that have been excluded from this survey. This survey only covered a literature-based analysis of the solutions to get a first conceptualization. Practical evaluations, like usability analyses, feature comparisons, and performance analyses between the different distributions, are helpful in gaining further insights.

References

[1] openSUSE Contributors, opensuse microos, 2023. URL: https://microos.opensuse.org/.

[2] Fedora, Fedora coreos documentation :: Fedora docs, 2023. URL: https://docs.fedoraproject.org/en-US/fedora-coreos/.

[3] Flatcar Project Contributors, Flatcar container linux | flatcar container linux, 2023. URL: https://flatcar-linux.org/.

[4] Bottlerocket OS, bottlerocket-os/bottlerocket: An operating system designed for hosting containers, 2023. URL: https://github.com/bottlerocket-os/bottlerocket.

[5] Sidero Labs, Inc., Talos linux, 2023. URL: https://www.talos.dev/.

[6] Fedora, Fedoracoreos, 2023. URL: https://getfedora.org/en/coreos?stream=stable.

[7] Amazon Web Services, Inc., Container host - bottlerocket - amazon web services, 2023. URL: https://aws.amazon.com/bottlerocket/.

[8] Cambridge University Press, Immutability | english meaning - cambridge dictionary, 2023. URL: https://dictionary.cambridge.org/dictionary/english/immutability.

[9] T. Rentsch, Object oriented programming, ACM SIGPLAN Notices 17 (1982) 51–57. doi:10.1145/947955.947961.

[10] B. Fitzgerald, N. Forsgren, K.-J. Stol, J. Humble, B. Doody, Infrastructure is software too!, SSRN Electronic Journal (2015). doi:10.2139/ssrn.2681904.

[11] K. Rothmund, Immutable linux distributionen mit linuxkit, Proceedings of the 2020 OMI Seminars Research Trends in Data Centre Operations, Selected Topics in Data Centre Operations, and Research Trends in the Internet of Things (PROMIS 2020) (2021) 1–10.

[12] Linuxkit, Linuxkit, 2023. URL: https://github.com/linuxkit/linuxkit.

[13] B. Tak, C. Isci, S. Duri, N. Bila, S. Nadgowda, J. Doran, Understanding security implications of using containers in the cloud, in: 2017 USENIX Annual Technical Conference (USENIX ATC 17), 2017, pp. 313–319.

[14] J. Castro, castrojo/awesome-immutable: A list of resources for people who want to investigate image-based linux desktops, 2022. URL: https://github.com/castrojo/awesome-immutable.

[15] Fedora, Configuring storage :: Fedora docs, 2023. URL: https://docs.fedoraproject.org/en-US/fedora-coreos/storage/.

[16] openSUSE Contributors, Portal:microos/design, 2022. URL: https://en.opensuse.org/Portal:MicroOS/Design.

[17] Flatcar Project Contributors, Flatcar container linux disk layout | flatcar container linux, 2023. URL: https://flatcar-linux.org/docs/latest/reference/developer-guides/sdk-disk-partitions/.

[18] LSB Workgroup, The Linux Foundation, Filesystem hierarchy standard, 2023. URL: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html.

[19] openSUSE Contributors, transactional-update, 2023. URL: https://kubic.opensuse.org/documentation/man-pages/transactional-update.8.html.

[20] Btrfs, btrfs wiki, 2023. URL: https://btrfs.wiki.kernel.org/index.php/Main_Page.

[21] Red Hat, Inc., rpm-ostree, 2023. URL: https://rpm-ostree.readthedocs.io/en/stable/.

[22] Flatcar Project Contributors, Performing manual flatcar container linux rollbacks | flatcar container linux, 2023. URL: https://flatcar-linux.org/docs/latest/setup/debug/

manual-rollbacks/.
[23] Sidero Labs, Inc., Upgrading talos linux | talos linux, 2023. URL: https://www.talos.dev/v1.3/talos-guides/upgrading-talos/.
[24] Fedora, Download fedora coreos, 2023. URL: https://getfedora.org/coreos/download?tab=metal_virtualized&stream=stable&arch=x86_64.
[25] Flatcar Project Contributors, Switching release channels | flatcar container linux, 2023. URL: https://flatcar-linux.org/docs/latest/setup/releases/switching-channels/.
[26] openSUSE Contributors, Portal:microos, 2022. URL: https://en.opensuse.org/Portal:MicroOS.
[27] Fedora, Auto-updates and manual rollbacks :: Fedora docs, 2023. URL: https://docs.fedoraproject.org/en-US/fedora-coreos/auto-updates/.
[28] Red Hat, Inc., Ignition - coreos/ignition, 2023. URL: https://coreos.github.io/ignition/.
[29] Flatcar Project Contributors, Ignition | flatcar container linux, 2023. URL: https://flatcar-linux.org/docs/latest/provisioning/ignition/.
[30] Fedora, Running fedora coreos directly from ram :: Fedora docs, 2023. URL: https://docs.fedoraproject.org/en-US/fedora-coreos/live-booting/.
[31] Canonical Ltd., cloud-init 22.4.2 documentation, 2023. URL: https://cloudinit.readthedocs.io/en/latest/.
[32] Sidero Labs, Inc., Editing machine configuration | talos linux, 2023. URL: https://www.talos.dev/v1.3/talos-guides/configuration/editing-machine-configuration/.
[33] C. Pahl, A. Brogi, J. Soldani, P. Jamshidi, Cloud container technologies: A state-of-the-art review, IEEE Transactions on Cloud Computing 7 (2019) 677–692. doi:10.1109/tcc.2017.2702586.
[34] L. Espe, A. Jindal, V. Podolskiy, M. Gerndt, Performance evaluation of container runtimes, in: Proceedings of the 10th International Conference on Cloud Computing and Services Science, SCITEPRESS - Science and Technology Publications, 2020. doi:10.5220/0009340402730281.
[35] Podman, Podman, 2023. URL: https://podman.io/.
[36] Fedora, Running containers :: Fedora docs, 2023. URL: https://docs.fedoraproject.org/en-US/fedora-coreos/running-containers/.
[37] Docker Inc., Docker: Accelerated, containerized application development, 2023. URL: https://www.docker.com/.
[38] Flatcar Project Contributors, Getting started with docker | flatcar container linux, 2023. URL: https://flatcar-linux.org/docs/latest/container-runtimes/getting-started-with-docker/.
[39] The Kubernetes Authors, Kubernetes, 2023. URL: https://kubernetes.io/.
[40] IEEE, IEEE recommended practice on software reliability, 2017. doi:10.1109/ieeestd.2017.7827907.
[41] C. Colman-Meixner, C. Develder, M. Tornatore, B. Mukherjee, A survey on resiliency techniques in cloud computing infrastructures and applications, IEEE Communications Surveys & Tutorials 18 (2016) 2244–2281. doi:10.1109/comst.2016.2531104.
[42] A. B. Bondi, Characteristics of scalability and their impact on performance, in: Proceedings of the 2nd international workshop on Software and performance, ACM, 2000. doi:10.1145/350391.350432.
[43] Fedora, Fedora coreos frequently asked questions :: Fedora docs, 2022. URL: https://docs.

fedoraproject.org/en-US/fedora-coreos/faq/.

[44] S. Böhm, G. Wirtz, Towards orchestration of cloud-edge architectures with kubernetes, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer International Publishing, 2022, pp. 207–230. doi:10.1007/978-3-031-06371-8_14.

[45] Google Cloud, Container-optimized os overview, 2022. URL: https://cloud.google.com/container-optimized-os/docs/concepts/features-and-benefits.

[46] Microsoft, Cbl-mariner documentation - cbl-mariner, 2022. URL: https://microsoft.github.io/CBL-Mariner/.

Author Index