Johannes Manner
Stephan Haarmann
Stefan Kolb
Nico Herzberg
Oliver Kopp

ZEUS 2021

13th ZEUS Workshop, ZEUS 2021,
Bamberg, Germany, 25–26 February 2021
Proceedings

Volume Editors

Johannes Manner
University of Bamberg, Distributed Systems Group
An der Weberei 5, DE-96049 Bamberg
johannes.manner@uni-bamberg.de

Stephan Haarmann
Hasso Plattner Institute, Business Process Technology
Prof.-Dr.-Helmert-Str. 2-3, DE-14482 Potsdam
stephan.haarmann@hpi.de

Stefan Kolb
JabRef Research
stefan.kolb@jabref.org

Nico Herzberg
Campeleon

Oliver Kopp
JabRef Research
oliver.kopp@jabref.org

Preface

In February 2021, we had the pleasure to organize the 13th edition of the ZEUS Workshop planned in Bamberg, Germany. Due to the ongoing covid-19 pandemic, the workshop is held virtually, giving us the chance to also invite our PC members which are one of ZEUS' success factors. Thanks a lot for your reviewing 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 thirteen submissions for presentation at the workshop. Each submission went through a thorough peer-review process and was assessed by at least five 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 will be generously sponsored by Camunda Services GmbH.

Bamberg, February 2021

Johannes Manner
Stephan Haarmann
Stefan Kolb
Nico Herzberg
Oliver Kopp

Organization

Steering Committee

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

Local Organizer

Robin Lichtenthäler	University of Bamberg
Sebastian Böhm	University of Bamberg

Program Committee Chairs

Stephan Haarmann	Hasso Plattner Institute, University of Potsdam
Johannes Manner	University of Bamberg

Program Committee

Saimir Bala	Vienna University of Economics and Business
Achim D. Bruckner	University of Exeter
Sebastian Böhm	University of Bamberg
Dirk Fahland	Eindhoven University of Technology
Manuel Fritz	University of Stuttgart
Matthias Geiger	University of Bamberg
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
Meiko Jensen	Kiel University of Applied Sciences
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
Adriatik Nikaj	Hasso Plattner Institute, University of Potsdam
Stefan Schulte	Vienna University of Technology
Jan Sürmeli	FZI Forschungszentrum Informatik, Karlsruhe
Stefan Winzinger	University of Bamberg
Michael Wurster	University of Stuttgart
Han van der Aa	Humboldt University of Berlin

Sponsoring Institutions

Camunda Services GmbH

Table of Contents

Fragment-Based Case Management Models: Metamodel, Consistency, and Correctness

Stephan Haarmann

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
stephan.haarmann@hpi.de

Abstract. Knowledge-intensive processes are inherently complex. Thus, modeling them is hard as the models have to capture various perspectives often using sub-models with hidden dependencies, e.g., the behavior of a process is constrained by cardinality constraints in a domain model. Yet, models are desirable as they are the primary tool in business process management to analyze, design, implement, and enact processes. We present a metamodel, consistency and correctness criteria for fragment-based case management. The criteria can i) be verified automatically and ii) used to assist modelers at design-time. Thereby, mistakes can be detected and even prevented during design, so model quality improves.

Keywords: Business Process Management, Process Modeling, Case Management

1 Introduction

Managing knowledge-intensive business processes is hard, and traditional business process management (BPM) is insufficient for this task [16]. While traditional business processes are highly structured, well-defined, and repetitive, knowledge-intensive ones are emergent, multi-variant, data-driven, and non-repeatable [3]. While highly structured processes are often modeled using imperative modeling languages, purely imperative models of knowledge-intensive processes are often perplexing.

Knowledge-intensive processes are executed by domain experts called knowlegde-workers, such as physicians, lawyers, and insurance clerks. The curse of the process is primarily determined by the decisions of the knowledge-workers, which are based on experience and case-specific information. Furthermore, knowledge-intensive processes are usually human-centered with little to no automation.

Novel modeling approaches fitted to knowledge-intensive processes have been proposed. Among them are case management approaches [1, 17]. In case management, knowledge workers gather, create, and maintain data. A case (process instance) evolves around this data. Every case has one central object, the case object (sometimes case folder). A case model describes both the data and the activities involved in a case as well as dependencies among them. Case management approaches can model knowledge-intensive processes more concisely than imperative languages. However, creating consistent and correct models can be difficult due to the inherent complexity and hidden dependencies. We propose a metamodel, consistency and correctness criteria for

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

fragment-based case management (fCM) [9]. fCM models are highly modular and capture data-centric processes concisely; however, they contain hidden dependencies, e.g., the behavior of the process is constrained by cardinality constraints in the data model. The criteria that we propose can be used to assist modelers to avoid mistakes.

In the next section, we introduce fCM with a metamodel and an example. Based on this, we present consistency and correctness criteria (Sect. 3). In Sect. 4, we discuss related work. Finally, we conclude and discuss our contribution (Sect. 5).

2 fCM Metamodel and Example

As depicted in Fig. 1, an fCM case model consists of a data model (*domain model*), a process model (*fragments*), and a goal specification called *termination condition* [9]. The data model consists of classes and (binary) *associations* among them[1]. Each class has an *object life cycle* (OLC), which defines the behavior of respective objects. A dedicated *case class* denotes the central object which is created exactly ones for each case.

An OLC is a finite state transition system. For each class, a set of *states* and state *transitions* is defined. This *object behavior* is instantiated by the process: knowledge workers execute activities that create data objects and update their states accordingly.

Activities are included in small control flow graphs, the so-called fragments. Activities read and write data objects that belong to a single input- and output-set, respectively.

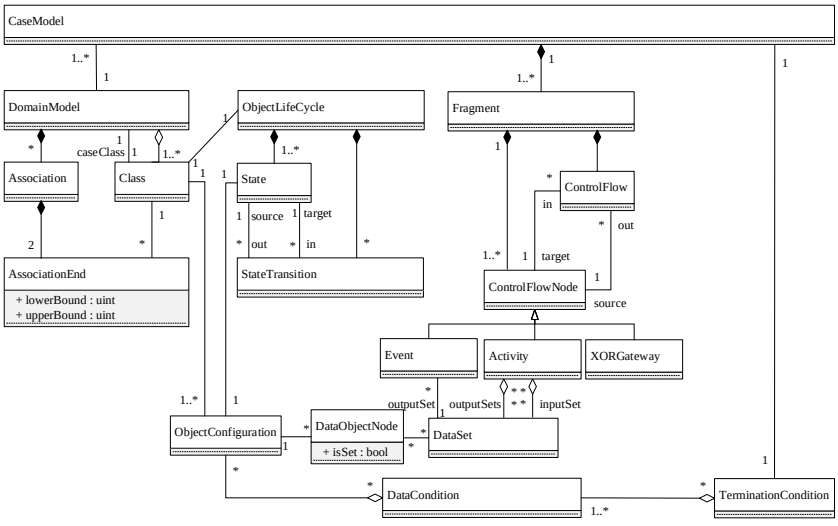


Fig. 1. fCM metamodel: a case model comprises a domain model including classes and associations, a set of fragments, and a termination condition.

[1] Due to space limitation, we do not consider *goal cardinality constraints*, which have been introduced in [6, 8].

Such objects are specified by *data object nodes*, which consist of an *object configuration* specifying the class and the state of the object and of an indicator (*isSet*, depicted as |||) showing whether multiple object's of the same type in the same state may be read.

Some fragments have start *events* marking the beginning of a new case. Such an event can create data objects. Also, a fragment can branch conditionally (*XOR-Gateway*).

While a fragment is similar to an imperative process model, all fragments operate on shared data. Thus, data requirements of activities can be used to model dependencies among fragments declaratively. At run-time, knowledge workers can instantiate and execute fragments repeatedly and concurrently if the activities' data-requirements permit.

A case that meets the termination condition can be closed by the knowledge workers.

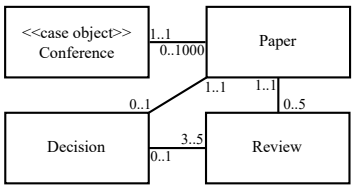


Fig. 2. Domain model of the paper submission and reviewing phase.

For an example, consider the paper submission and reviewing at an academic conference. The domain model Fig. 2 comprises the classes `Conference`, `Paper`, `Review`, and `Decision`. `Conference` is the case class. For each conference, multiple papers can be submitted; for each paper, multiple reviews can be created; for each paper, a decision is made based on at least three reviews.

Figure 3 shows the OLCs for the classes in Fig. 2. All but one OLC are described by a sequence of states, but OLCs can branch in case of alternative state progression and even contain disconnected subgraphs in case of alternative initial states. A decision object can be in one of the two alternative states *accepted* and *rejected*. OLCs do not define initial or final states. Since activities, create and change data objects, initial and final states are encoded in the process behavior. Adding them to the OLCs would add redundancy but provide only little value.

The case behavior is defined by a set of fragments (see Fig. 4). The example has one start event in fragment *f1*. When the start event occurs, a new case is started; the conference object is created; and activity "open submission" is enabled. Afterwards, the requirements of "submit paper" in fragment *f2* are satisfied. It can be executed repeatedly. Eventually, the knowledge workers perform "close submission" (*f1*) changing the state of the conference object and disabling fragment *f2* consequently. It also changes all papers

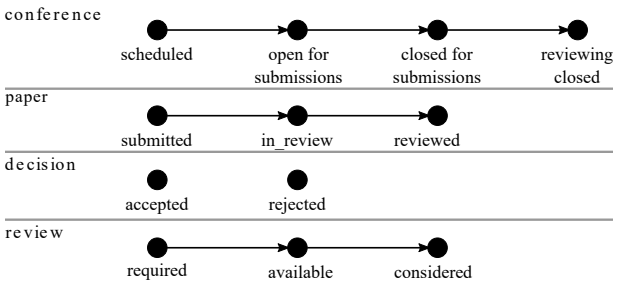


Fig. 3. Object life cycles for the classes in the domain model (Fig. 2)

to state *in review* (shown by the set indicator |||). Fragment *f3* can hence be executed to assign reviewers to papers. From the domain model, we know that each paper has at most 5 reviews. Therefore, fragment *f3* can at most be executed 5 times for each paper. As soon as a reviewer has been assigned, the review can be created (fragment *f4*). While there is a dependency between the fragments *f3* and *f4*, a review can be created before, after, or during the assignment of other reviewers. The order is determined by the knowledge workers. Similarly, activity "decide on paper" (fragment *f5*) can be executed as soon as all reviews assigned to a particular paper have been created. The activity has three possible outcomes: if there are less than 5 reviews for the paper, an additional reviewer may be assigned (output set {review[required]}; if there are at least 3 reviews, the paper may be rejected (output set {paper[rejected],reviews[considered],paper[reviewed]}) or accepted (output set {paper[accepted],reviews[considered],paper[reviewed]}). The different output sets are not part of the visual notation. Once all papers of the conference are in state *reviewed*, the reviewing can be closed (fragment *f1*).

The termination condition `conference[reviewing closed]` is satisfied after fragment *f1* has been executed completely. The case can be closed.

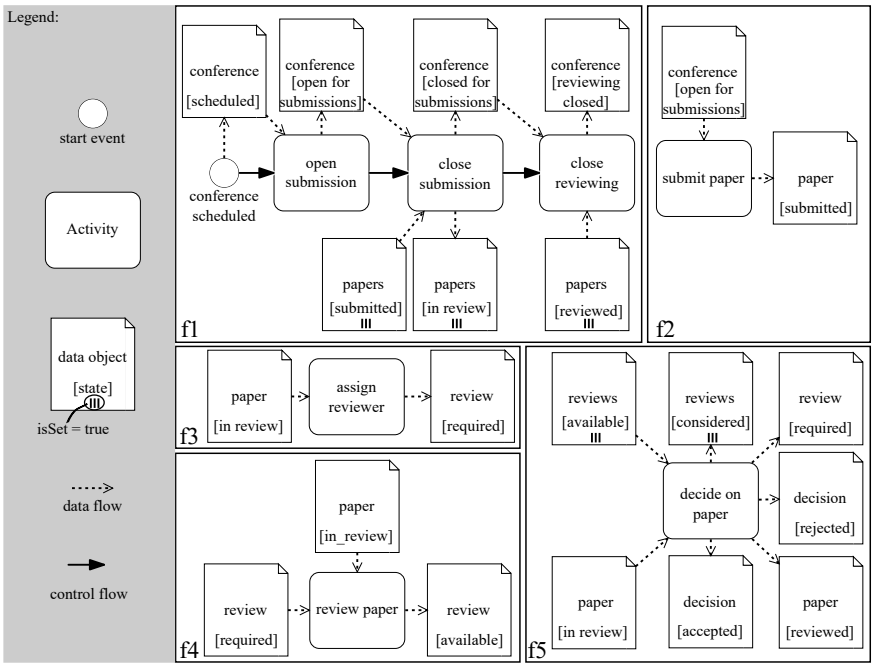


Fig. 4. Fragments for the paper submission and reviewing at an academic conference. Fragment f1 captures the progression of the conference. Fragment f2 handles the paper submission, f3 the assignment of reviewers, f4 the creation of reviews, and f5 the decision whether a paper is accepted, rejected, or an additional review is required.

3 Consistency and Correctness Criteria

As imminent from the example, all parts of the case model play together during a case. Fragments are connected through shared data, whose structure and behavior is modeled by the domain model and OLCs. The knowledge workers choose from enabled activities to progress the case towards the termination condition. Therefore, it is important that i) the parts are correctly modeled and ii) consistently integrated. In this section, we briefly sketch some structural correctness and consistency criteria that must be satisfied.

Assumptions. While the domain model focus on structuring the data, associations have behavioral implications. For one, they may define a partial order in which objects must be created [15]. In the example, every review requires a paper, but a paper may have no reviews (yet). Consequently, the review cannot be created before the paper. However, such an order can only be inferred if one object depends on another. Therefore, we require that all associations are existential: at least one of the corresponding lower bounds must be positive. Furthermore, we only allow one association between a pair of classes and disallow many-to-many associations (one of the association's upper bounds must be 1). If these assumptions are satisfied, new associations are only established when new objects are created, e.g., activity "assign reviewer" reads a paper and creates and associates a review. If the assumptions are violated, the domain model can be reified: new classes can be introduced in place of the violating associations.

Additionally, we make assumptions about the structure of fragments. We assume that fragments are acyclic. This does not limit the expressiveness since loops can be resolved into repeatable fragments. For example, in a purely imperative process model, activity "submit paper" would be part of a loop instead of a fragment. We furthermore assume that each fragment is either initial or non-initial. This means they either start with an event (e.g., fragment *f1* in Fig. 4) or with an activity (all fragments but *f1*)—never both.

Consistent I/O Behavior. Since fCM models consist of data and behavioral parts, most dependencies apply to the I/O behavior of activities. Correct I/O behavior depends on the domain model, the object life cycles, other activities, and even the termination condition.

First, all data requirements must be *satisfiable*. Therefore, some activity must produce the object configuration (object in a certain state) required by other activities or the termination condition. This means, each data configuration used in a data object node or a data condition should be referred by a data object node that takes part in at least one output set. For example, the object configuration `conference[reviewing closed]` must be written by at least one activity, i.e., "close submission". Assuming the termination condition would instead be `conference[proceedings published]` and the state would be an allowed successor of *reviewing closed*, the termination condition would be insatiable since no activity writes the conference object in the respective state.

Furthermore, for each output set of an activity must exist a respective input set so that the combination conforms to the constraints of the OLCs and the domain model.

A valid input-output-set combination is *OLC conform* [9]: all the subsumed state transitions must be present in the OLCs. If an activity "skip submission" changes the state of conference from *scheduled* to *closed for submission*, the OLC would be violated. This property is called *object life cycle conformance*.

Next, each object that is created requires a specific *context*, a set of objects it depends on. The required context is defined by the existential associations. In the example, the decision requires a paper and three to five reviews. This context must be provided by the input-output-combination. The required objects must either be read or co-created. If activity "assign reviewer" would not read a paper object, the requirements for the review would be violated. If execute anyway, the created review object would violate the cardinality constraints specified in the domain model (cf. Fig. 2).

What about the reviews required for a decision? According to the cardinality constraints, a decision existentially depends on three to five reviews. In such cases a set of objects (`isSet=true` visualized by |||) must be read, e.g., a set of at least three reviews must be read to create a decision. We call this *mandatory batch behavior*.

Finally, if a set of objects is read, it must be clearly defined. Therefore, each input set that contains a data object node with `isSet=true` must also contain a data object node with `isSet=false` for an associated class. The respective object is used to determine the set of objects that is co-read when executing the respective activity. In the example, "decide on paper" reads all reviews that belong to the paper. Without the paper, the set cannot be determined from the context.

The presented criteria are not domain specific, i.e., they do not only apply to the example fCM model but to all possible fCM models. Any case model that satisfies all criteria presented in this section is *structural consistent*. The structure of one part (e.g., the fragments) does not contradict the structure of other parts (e.g., the domain model).

4 Related Work

The fragment-based case management [9] approach is a production case management approach based on [11]. A metamodel for fCM has been proposed in [5]. The metamodel is close to [9] and includes elements that are fix for all models, such as generic life cycles for activities. Additionally, extensions have been proposed that consider associations [7] and cardinality constraints [6, 8]. The metamodel presented in this paper is the first fCM metamodel considering associations and cardinality constraints.

Besides fCM there are other approaches combining information from process and data modeling. Combi et al. [2] and Meyer et al. [12] combine data models, i.e., UML class diagrams, and process models, i.e., BPMN diagrams. [2] describes and detects inconsistencies while [12] derives SQL-queries for enactment. Montali et al. [13] introduce DB-nets—a Petri net-based formalism for modeling data-base accessing processes that adhere to data constraints (e.g., primary key, foreign key, and cardinality constraints). Therefore, they introduce a transaction mechanism to the processes. Ghilardi et al. [4] present catalog-nets to formally model processes with access to read only data-bases. While these approaches elaborate the connection between data and process, they are purely imperative and not suited for knowledge-intensive processes. However, DB-nets and catalog nets are interesting formalisms, which may be suited to formalize fCM's semantics and to define/verify behavioral correctness notions.

Other case management approaches exist. The two most prominent ones are the Guard Stage Milestone [10] approach and the derived Case Management Model and Notation [14] standard. Both approaches arrange activities into stages and compose

behavior based on (data-based) pre- and post conditions. While the approaches assume a data model (called information model), they do not specify how it is modeled and integrated into the process specification.

5 Conclusion

Case models capture knowledge-intensive processes, which are often data-driven, multi-variant, and non-repeatable. Consequently, models become quite complex as they must integrate data and flexible processes. In this paper, we present a metamodel, consistency and correctness criteria for fCM models. All fCM models must satisfy these criteria.

Future work may elaborate them. First, formal definitions should be provided, e.g., using first-order logic or the object constraint language. Such definitions can be the base for implementing i) a verification tool that detects violations of the criteria and ii) a modeling tool that support case designers. Such a tool can highlight violations and offer auto-completion/correction. For example, when a case designer models an activity that creates a data object, all required objects (according to the domain model) can be added to the activity's input set if they are created by other activities or output sets otherwise.

Structural correctness and consistency is important and verification is computational in-expensive compared to state space analysis. However, such criteria cannot guarantee correct behavior. In future work, we want to investigate behavioral correctness criteria that apply to case models. A first example is weak termination: in the initial state, it should be possible to reach a state that satisfies the termination condition.

Furthermore, fCM does not capture all aspects of a case. Knowledge-intensive processes are also about knowledge workers, their rights, capabilities, and collaborations among them [3]. In the future, the fCM metamodel and language may be extended to account for the user perspective. The correctness and consistency criteria may subsequently be refined to consider the additional information.

Nevertheless, we believe that the presented metamodel and criteria can lead to tool support for fCM that may ultimately improve the accessibility of the fCM approach.

References

1. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. 53(2), 129–162 (2005)
2. Combi, C., Oliboni, B., Weske, M., Zerbato, F.: Conceptual modeling of processes and data: Connecting different perspectives. In: Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings. pp. 236–250 (2018)
3. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. J. Data Semant. 4(1), 29–57 (2015)
4. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Petri nets with parameterised data - modelling and verification. In: Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings. pp. 55–74 (2020)
5. Gonzalez-Lopez, F., Pufahl, L.: A landscape for case models. In: Enterprise, Business-Process and Information Systems Modeling - 20th International Conference, BPMDS 2019, 24th International Conference, EMMSAD 2019, Held at CAiSE 2019, Rome, Italy, June 3-4, 2019, Proceedings. pp. 87–102 (2019)

6. Haarmann, S., Montali, M., Weske, M.: Technical report: Refining case models using cardinality constraints. CoRR abs/2012.02245 (2020), https://arxiv.org/abs/2012.02245
7. Haarmann, S., Weske, M.: Correlating data objects in fragment-based case management. In: Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings. pp. 197–209 (2020)
8. Haarmann, S., Weske, M.: Data object cardinalities in flexible business processes. In: Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers. pp. 380–391 (2020)
9. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings. pp. 38–54 (2016)
10. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.F.T., Hobson, S., Linehan, M.H., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculín, R.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers. pp. 1–24 (2010)
11. Meyer, A., Herzberg, N., Puhlmann, F., Weske, M.: Implementation framework for production case management: Modeling and execution. In: 18th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2014, Ulm, Germany, September 1-5, 2014. pp. 190–199 (2014)
12. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings. pp. 171–186 (2013)
13. Montali, M., Rivkin, A.: From DB-nets to coloured Petri nets with priorities. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. pp. 449–469 (2019)
14. (OMG), O.M.G.: Case management model and notation (CMMN) (December 2016), https://www.omg.org/spec/CMMN
15. Snoeck, M.: Enterprise Information Systems Engineering - The MERODE Approach. The Enterprise Engineering Series, Springer (2014)
16. Swenson, K.D.: Position: BPMN is incompatible with ACM. In: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers. pp. 55–58 (2012)
17. Swenson, K.D.: State of the art in case management - 2013 (2012), https://www.aiim.org/PDFDocuments/CaseManagement2013.pdf

All links were last followed on January 18, 2021.

Towards Decision Management for Robotic Process Automation

Simon Siegert and Maximilian Völker

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
simon.siegert@student.hpi.de
maximilian.voelker@hpi.de

Abstract. Robotic process automation (RPA) is a rapidly growing technology for automating digital processes. While RPA allows the automation of common tasks performed on a computer, there are only rudimentary possibilities to represent decisions in RPA models. Especially workflows that involve more than simple yes-no questions quickly result in confusing models, which are difficult to understand and maintain. To overcome these issues, an integration of an established decision management approach, Decision Model and Notation (DMN), into RPA is proposed and motivated in this paper.

Keywords: Robotic Process Automation, RPA Lifecycle, Decision Management

1 Introduction

Automation based on process models has long been an area of interest in business process management (BPM), for both the research and the enterprise world [13]. Robotic process automation (RPA) is a novel automation approach employing software robots to automate undemanding tasks on the computer, that recently gained more attention in research [2]. Whereas BPM systems target organization-wide processes, RPA automates sequences of tasks that employees perform locally on their computers by imitating the employee's behavior on the graphical user interface level [6, 12], making it a useful addition to BPM [14].

To enable business users and employees to create their own automations, many current RPA vendors focus on the visual representation and modeling of RPA robots [3]. However, a preliminary analysis of different RPA products (UIPath, Blue Prism, Automation Anywhere, and Automagica) revealed disadvantages of these approaches, especially with respect to the design of decisions. A majority of the representation types examined only supported simple $if/else$- or $switch/case$ constructs to model decisions. Since each decision branch must be represented explicitly, the RPA models become complex, especially for more elaborate decisions. This may lead to decision-intensive processes being discarded for automation using RPA due to the excessive modeling effort and decreased maintainability, although they might be well suited for automation.

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

This paper outlines a possible solution to overcome the issue of representing complex decisions in RPA models. Based on the mature standard for decision management, DMN [11], a way of integrating decisions in RPA robots is delineated, maintaining the focus on business users and intuitive usability.

2 Related Work

In BPM, similar problems regarding the modeling of decisions in workflows have been reported, such as complex, nested structures which are hard to maintain and understand [1]. For one of the associated modeling notations, BPMN [10], the problem was addressed by separating the decision logic and the control flow using DMN (Decision Model and Notation) [5, 7]. DMN allows the modeling of decision requirements and the specification of the underlying decision logic in the form of a decision table [11]. In BPMN models, DMN is used to encapsulate the previously branched and nested decisions into a single new decision activity. This allows decision-intensive business process models to be presented in a clear and compact way. Even approaches to extract the decision logic from process models and replace it with DMN have been proposed [4].

However, although current graphical RPA approaches suffer the same problem, it has, to the best of our knowledge, not yet been addressed in research. Nevertheless, since RPA software is known to be highly rule-based [12], it is expected to benefit from the integration of a proven technique for data-based and rule-based decisions, such as DMN.

3 Motivation

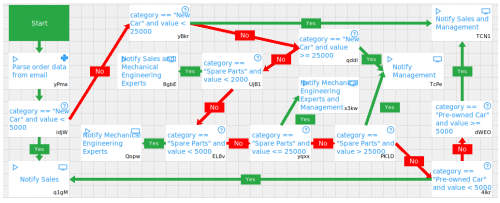


Fig. 1. Scenario implemented with explicitly modeled decisions (intentionally not readable)

Consider the following scenario: In a car dealership, orders placed by employees for the company are to be released. To do this, they send their orders to a secretary, who decides which departments in the company are eligible to approve an order. Since orders can have different costs and categories (such as 'New Car' or 'Spare Parts'), and depending on this can have one or more parties in the company authorized to review them (e.g., 'Management' or 'Sales Department'), the decision has a certain degree of complexity.

Implementing the scenario with RPA is a viable option here, as the costs and category extraction, the decision, and the email notifications can be automated using common RPA features. Figure 1 shows a possible implementation with

the open source version of the RPA software Automagica[1], demonstrating that explicit decision modeling can lead to an unreadable spaghetti-like model.

Here, the explicit but necessary modeling of the decision with the different options of responsibility to decide on orders as well as the different combinations of the input variables 'cost' and 'order category' leads to a nested decision tree that is difficult to understand.

4 Integrating RPA with DMN

The research proposal is to apply elements from decision management of BPM to the modeling and implementation of RPA systems. Hiding the complexity of a decision using a decision engine makes the RPA process model more understandable. For example, Figure 2 shows a prototypical implementation of a software robot that executes the same program logic as the model in Figure 1, but requires much less modeling elements as the decision resides in a decision table that is evaluated by a decision engine in the task "Decide on responsible party". The

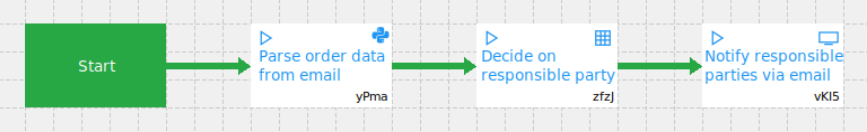


Fig. 2. Order review scenario modeled with RPA and DMN

decision takes place within a DMN decision table, which is not shown here. To create the decision table, however, further modeling effort is necessary, especially for users who are not familiar with DMN. Thus, two different modeling languages have to be learned to create RPA process models.

The research approach is to investigate for all phases of the RPA lifecycle, as defined by König et al. [9] and Jimenez-Ramirez et al. [8], how decisions in RPA bots can be supported by DMN and how each lifecycle phase needs to be adapted compared to existing approaches. The selection of suitable business processes for automation, necessary capabilities of modeling tools and requirements for RPA architectures (e.g., local decision engine versus external decision service) as well as their implementation will be compared and discussed. Communication flows between individual software components, verification of correct behavior and efficient operation are also to be considered. In order to demonstrate the feasibility and to evaluate the benefits of integrating DMN in robotic process automation, several prototypes addressing the different architecture types (e.g.,

[1] `https://github.com/automagica/automagica/tree/ae8a1846f23df6497e725c8db198b4420da82f12` (latest open source version)

decision engine embedded in the RPA robot or connection to an external decision engine) will be implemented and compared.

5 Conclusion

The proposed introduction of a dedicated decision component for RPA is applicable for rule-based and data-focused RPA processes. Although the decision component should not and cannot be applied to every decision in RPA processes (simple decisions can still be mapped using conventional means such as $if-else$), more complex and data-driven processes are expected to benefit from the use of more sophisticated decision management. Overall, it may lead to better monitoring, easier updating, and higher overall comprehensibility of RPA models and executions compared to explicitly modeled decision trees, which highlights the need for further research in this direction. Future work could analyze whether and how DMN in RPA could benefit from the integration of machine learning technologies and conduct a user study to analyze the trade-off between reduced model complexity and additional effort due to an additional modeling language.

References

1. van der Aa, H., Leopold, H., Batoulis, K., Weske, M., Reijers, H.A.: Integrated process and decision modeling for data-driven processes. In: Reichert, M., Reijers, H.A. (eds.) Business Process Management Workshops. pp. 405–417. Springer International Publishing, Cham (2016)
2. Van der Aalst, W.M., Bichler, M., Heinzl, A.: Robotic process automation (2018)
3. Aguirre, S., Rodriguez, A.: Automation of a business process using robotic process automation (rpa): A case study. In: Figueroa-García, J.C., López-Santana, E.R., Villa-Ramírez, J.L., Ferro-Escobar, R. (eds.) Applied Computer Sciences in Engineering. pp. 65–71. Springer International Publishing, Cham (2017)
4. Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M.: Extracting decision logic from process models. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) Advanced Information Systems Engineering. pp. 349–366. Springer International Publishing, Cham (2015)
5. Biard, T., Le Mauff, A., Bigand, M., Bourey, J.P.: Separation of decision modeling from business process modeling using new "decision model and notation" (dmn) for automating operational decision-making. In: Camarinha-Matos, L.M., Bénaben, F., Picard, W. (eds.) Risks and Resilience of Collaborative Networks. pp. 489–496. Springer International Publishing, Cham (2015)
6. Flechsig, C., Lohmer, J., Lasch, R.: Realizing the full potential of robotic process automation through a combination with bpm. In: Bierwirth, C., Kirschstein, T., Sackmann, D. (eds.) Logistics Management. pp. 104–119. Springer International Publishing, Cham (2019)
7. Hasić, F., Devadder, L., Dochez, M., Hanot, J., De Smedt, J., Vanthienen, J.: Challenges in refactoring processes to include decision modelling. In: Teniente, E., Weidlich, M. (eds.) Business Process Management Workshops. pp. 529–541. Springer International Publishing, Cham (2018)
8. Jimenez-Ramirez, A., Reijers, H.A., Barba, I., Del Valle, C.: A method to improve the early stages of the robotic process automation lifecycle. In: International Conference on Advanced Information Systems Engineering. pp. 446–461. Springer (2019)
9. König, M., Bein, L., Nikaj, A., Weske, M.: Integrating robotic process automation into business process management. In: Asatiani, A., García, J.M., Helander, N., Jiménez-Ramírez, A., Koschmider, A., Mendling, J., Meroni, G., Reijers, H.A. (eds.) Business Process Management: Blockchain and Robotic Process Automation Forum. pp. 132–146. Springer International Publishing, Cham (2020)
10. Object Management Group (OMG): Business Process Model and Notation (BPMN) Specification, Version 2.0. `https://www.omg.org/spec/BPMN/2.0/` (2011)
11. Object Management Group (OMG): Decision Model and Notation (DMN) Specification, Version 1.3. `https://www.omg.org/spec/DMN/1.3/` (2019)
12. Syed, R., Suriadi, S., Adams, M., Bandara, W., Leemans, S.J., Ouyang, C., ter Hofstede, A.H., van de Weerd, I., Wynn, M.T., Reijers, H.A.: Robotic process automation: Contemporary themes and challenges. Computers in Industry 115, 103162 (2020)
13. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, third edn. (2019)
14. Willcocks, L.P., Lacity, M., Craig, A.: The IT function and robotic process automation. The Outsourcing Unit Working Research Paper Series (15/05), The London School of Economics and Political Science, London, UK (2015)

Towards Real-Time Progress Determination of Object-Aware Business Processes

Lisa Arnold, Marius Breitmayer and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{lisa.arnold, marius.breitmayer, manfred.reichert}@uni-ulm.de

Abstract. To stay competitive, companies need to continuously improve and evolve their business processes. In this endeavour, business process optimisations and improvements are key elements. In particular, the monitoring of business processes enables the early discovery of problems and errors already during process enactment. Two approaches can be pursued to achieve this: real-time, also called online monitoring, and offline monitoring. A subtask of real-time monitoring is to determine the current progress of a business process, which is particularly challenging if the latter is composed of loosely coupled, smaller processes that interact with each other, like object lifecycle processes in data-centric approaches to BPM, which result in large process structures. This position paper discusses the challenges of determining the progress of such object-aware processes in real-time and defines research questions that need to be investigated in further work.

Keywords: object-aware business process, process monitoring, progress determination, online/real-time monitoring

1 Introduction

Object-aware business processes consist of interacting objects whose relations are defined in a relational process structure [1]. Each object has attributes and a lifecycle process describing its behaviour [2], whereas coordination processes structure and control the interactions between multiple lifecycle processes, i.e., the overall business process [3]. Lifecycle processes comprise states (including exactly one start and at least one end state), and state transitions. Each state, in turn, consists of connected steps, where each step corresponds to an attribute update operation. Based on this information, electronic forms are auto-generated during lifecycle execution for each state. As an example, Figure 2 shows a lifecycle process with one decision step. In general, an object-aware business process is composed of multiple lifecycle processes of the same or different type, resulting in a dynamically evolving process structure during runtime. In this context, a variety of configurations and constellations of object-aware processes (i.e., lifecycle and coordination processes), running concurrently to each other, becomes possible [3]. Additionally, there are coordination constraints between objects that require interactions between the corresponding lifecycles. Due to

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

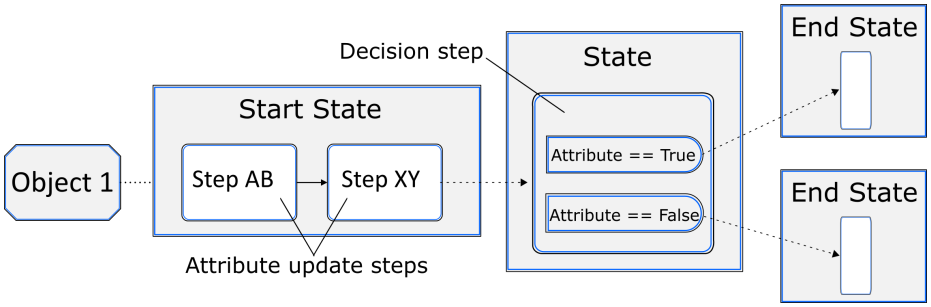


Fig. 1. Lifecycle structure in step-based view

this high complexity, there is no intuitive solution for monitoring and measuring the progress of an object-aware business process and the corresponding process structure respectively.

Several challenges to determine and define progress metrics in object-aware process management exist. First, no known measures for object-aware progress exists. Second, progress can be interpreted in different ways. Often, it is described as fundamental improvement through significant changes of the current status. In general, however, there is no broadly accepted definition of the term *progress*. Third, *measuring points* for determining the progress of an object-aware business process are missing. Fourth, to determine the progress the current execution status as well as the path yet to be taken (including all routing decisions) are required. Fifth, a time delay in the calculation of the progress or problems in estimating the execution path can occur. Thus, we need to define metrics for determining the progress of an object-aware process in all possible constellations. In addition, the progress of many individual processes (i.e., lifecycles and coordination process) needs to be properly merged to determine overall progress.

2 Related Work

In [4], an approach to measure the progress of activity-centric business models is discussed. An approach for monitoring processes and predicting their progress with data state transition events is presented in [5]. In turn, [6] improves progress in activity-centric processes using object state transition [5]. By contrast, no approaches exist for determining progress in object-aware business process management. Progress measurement of processes in other fields than BPM can be found, for example, in construction [7], software engineering [8], and software management [9].

3 Research Questions

As a first step towards the online progress calculation of an object-aware process, it must be possible to determine the progress of any snapshot of this process.

Moreover, the total progress of the object-aware process (i.e., the overall business process) is determined by the progress of its individual lifecycles. In a nutshell, any approach for determining the progress of an object-aware process needs to answer the following research questions:

Research Question 1	How can the progress of a single lifecycle process with its state-based view form be determined?
Research Question 2	How can the progress of the processing of a single state within a lifecycle process be measured?
Research Question 3	How can the progress of multiple, interacting (i.e., interrelated) lifecycles be determined?
Research Question 4	How does a coordination process affect the progress of an object-aware business process?

Research Question 1 considers single lifecycle processes in their abstracted (i.e., state-based) view, whereas Research Question 2 deals with determining the intra-state progress. Answering Research Questions 1 and 2 will enable us to fully determine the progress of a lifecycle. For example, for every lifecycle depicted in Figure 2, a progress between 0 (i.e., instantiation of a lifecycle) and 100 (i.e., execution of an end state and lifecycle completion) may be assigned. Research Question 3 extends progress determination to a full relational process structure consisting of multiple interacting lifecycle processes (see Figure 2). The latter form a large process structure whose overall progress needs to be determined. Research Question 4 considers the coordination (depicted as dashed arrows in Figure 2) of the relational process structure to refine Research Question 3. With Research Question 4 the total progress of object-aware business processes can be addressed. All four research questions need to be answered to be able to determine the total progress of an object-aware business process.

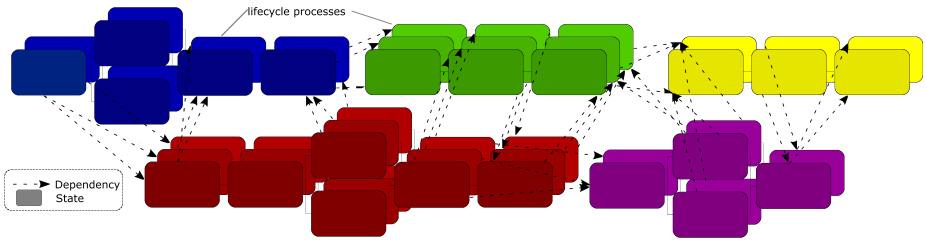


Fig. 2. Complex structure of an object-aware business process

4 Conclusions

This position paper discussed the challenges to be tackled when determining the progress of object-aware processes and the potentially very large lifecycle process structure emerging during their execution. As a major benefit of being able to determine the progress of object-based business processes, the real-time monitoring of the underlying large process structures becomes possible. Although just-in-time business scenarios are common in many business areas, contemporary business process management tools do not provide a sufficient and timely measurement of the progress of the emerging process structures. Due to the complex structure, research questions were addressed, which should be clarified in further work. Thereby, approaches based on graph theory can be considered for discussing the presented research questions. The results can be refined using event log data in combination with approaches from probability theory and machine learning. In this way, the results of decision steps as well as the total workload necessary of a process can be predicted to determine the progress more accurately.

Acknowledgement. *This work is part of the ZAFH Intralogistik, funded by the European Regional Development Fund and the Ministry of Science, Research and Arts of Baden-Württemberg, Germany (F.No. 32-7545.24-17/3/1).*

References

1. Steinau, S., Andrews, K. & Reichert, M. *The relational process structure* in *International Conference on Advanced Information Systems Engineering* (2018), 53–67.
2. Steinau, S., Andrews, K. & Reichert, M. *Executing Lifecycle Processes in Object-Aware Process Management* in *Data-Driven Process Discovery and Analysis* (Springer, 2019), 25–44.
3. Steinau, S., Künzle, V., Andrews, K. & Reichert, M. *Coordinating business processes using semantic relationships* in *IEEE 19th Conf on Business Informatics (CBI)* (2017), 33–42.
4. Koschmider, A., Vara, J. L. d. l. & Sánchez, J. *Measuring the progress of reference model-based business process modeling* in *INFORMATIK 2010* (), 218–229.
5. Herzberg, N. & Meyer, A. Improving process monitoring and progress prediction with data state transition events. *Hasso Plattner Institute at the University of Potsdam* (2013).
6. Herzberg, N., Meyer, A. & Weske, M. Improving business process intelligence by observing object state transitions. *Data & Knowledge Engineering,* 144–164 (2015).
7. Zhang, X. *et al.* Automating progress measurement of construction projects. *Automation in Construction,* 294–301 (2009).

8. Sommerville, I. Software engineering 9th Edition. *ISBN-10,* 18 (2011).
9. Kerzner, H. *Project management: a systems approach to planning, scheduling, and controlling* (John Wiley & Sons, 2017).

Towards a Framework for Data Enhanced Process Models in Process Mining

Jonas Cremerius

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
jonas.cremerius@hpi.de

Abstract. Understanding and improving business processes have become important steps towards success for organizations. Getting insights about a process is not only based on the control flow, but also on the data generated within the process. Today, almost every process step generates data, especially in the health care domain. So far, most analyzes inside a process model are limited to time analyzes and identification of decision logic. However, various attributes can be linked to events, such as the number of abnormal lab values derived from a lab test, which could be displayed in the process model. This can help to explore the process with domain experts, where they can choose the attributes of interest for each event and observe their influence on the process, i.e. the influence of abnormal lab values on the treatment process. Therefore, the interplay of event attributes and the control flow can be observed directly in the process model.

Keywords: Process Mining · Process Enhancement · Process Model Extension

1 Introduction

Business process models play a central role in exploring and analyzing the organization's business processes. With the help of process mining, process models can be derived from real-world process execution data [3]. Process mining is often conducted in the healthcare sector, as hospitals are becoming increasingly aware of the need to improve their processes [8]. Despite the increasing availability of data, adequate support for displaying or analyzing event attributes in a discovered process model is still lacking. So far, process enhancement inside the discovered process model is limited to time analyzes and decision logic, whereas organizational aspects are separately analyzed [15]. This is also represented in today's process mining tools. Fluxicon Disco[1] provides time analyzes only, whereas Lana Labs[2], Celonis[3], and ProM[4] do not provide attribute analyzes inside the process model. Only Process Diamond[5] allows displaying one

[1] https://www.fluxicon.com/disco/
[2] https://www.lanalabs.com/
[3] https://www.celonis.com/
[4] https://www.promtools.org/
[5] https://www.processdiamond.com/

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

attribute, which must be the same for all events. In several treatment processes, such as the diagnosis and treatment of heart failure, a large amount of data is generated [1]. Example events include "Analyze Lab Values" and "Treat Patient in Intensive Care Unit (ICU)". For both events, a different set of attributes is produced, such as the number of abnormal lab values and the Glasgow Coma Scale (GCS). Considering the effect of these attributes on a process may provide insights about a certain diagnostic or treatment. As current process mining tools are not capable of exploring a process in respect to attribute values, we propose a respective functionality that enables business analysts and domain experts to identify conspicuous behavior not only in the control flow, but also in the attribute values.

2 Related Work

Process Enhancement is the extension or improvement of an existing process model using information about the actual process recorded in some event log [3]. There exist several approaches which analyze the activity duration and waiting time in a process model to identify, for example, bottlenecks [10,14]. Additional analyzes inside the process model include identification of constraints at decision points [7, 11]. Further, Process Enhancement involves the organizational analyzes by looking at the assignment of human resources to process activities [12]. Enhancement is also conducted offside the process model where, for instance, different process characteristics are correlated with each other [5,13]. Case and event attributes have been used in filtering process variants and clustering of traces. For instance, [4] clusters traces to discover process variants by different case and event attributes, such as age and the body part of an image analyzes.

3 Research Objective

None of the approaches mentioned above looked into displaying event attributes directly in the process model. So far, they are limited to time analyzes and identification of decision points. Our research aims to fill this gap and provide a framework to link event attributes with their respective event in the process model. Today's data sets, such as MIMIC-IV, include various event attributes, such as the number of abnormal lab values [2]. As processes can get complex, different attributes might be interesting for the events. An example process is displayed in Fig. 1, which illustrates a simple treatment process. The events have their attributes attached in the process model. The first two events "Perform X-Ray" and "Perform CT" show the frequency of findings in specific body regions (heart and lung). After that, the lab values are analyzed, which show the frequency of abnormal lab values observed for each lab value. Then, the treatment is conducted, which can happen in the Intensive Care Unit (ICU) or Cardiology, where the mean Glasgow Coma Score (GCS) is shown. This process can be similar for several diseases on this abstraction level. Therefore, the insights

regarding the control flow might be limited. However, the event attributes can be different, as the glucose level is more interesting for type 2 diabetes and the creatinine level for kidney disease [6,9]. This could help to assess how meaningful analyzing a specific lab value for several diseases is. Furthermore, the effect of an attribute on the process can be explored with the help of a process model. For example, if the process of patients with an abnormal glucose level is of interest, one could just click on the attribute, triggering filtering according to the lab value. Then, not only the change in the process flow, but also in the event attributes can be seen, such as the mean GCS or the frequency of findings in the heart region of an image analyzes, which might be different for patients with an abnormal lab value. This could reveal novel insights, as one might not have thought that patients with an abnormal lab value also have a high prevalence of findings in the heart region of an X-ray.

We want to enable this kind of analyzes in the process model, which could lead to a more comprehensive exploration of processes together with a domain expert. The framework defines how different types of attributes are displayed in the process model and which computations can be performed on them, such as minimum, maximum, or mean. As process attributes can be different depending on the application context, we want to enable the domain experts to choose the attributes of interest. Nevertheless, an attribute recommendation system could be implemented, which helps to choose the appropriate attributes for an activity based on machine learning or descriptive statistics. Additionally, the framework could help to highlight events sharing the same attributes, such as different lab values or medical imaging techniques. Therefore, the following research questions need to be answered:

- How can event attributes be displayed in the process model (categorical vs. continuous variables)?
- How can the framework help to gain new insights about the process (detection of process variants, dependencies between attributes, etc.)?

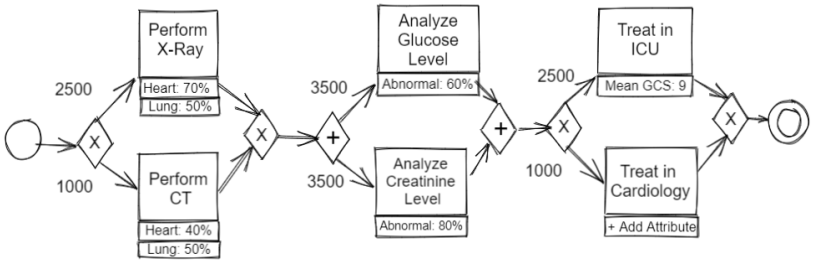


Fig. 1. Process model with data attributes displayed for each event. CT, Computed Tomography; ICU, Intensive Care Unit; GCS, Glasgow Coma Score

4 Conclusion

This position paper discusses the need for looking at the inclusion of event attributes directly in the process model. With the increasing data availability, a more comprehensive view of the process is possible and different process variants can be explored. As process models can be used as a means of communication between process analysts and domain experts, incorporating event data in the model has the potential to improve that communication by illustrating the control flow and event attributes in one place.

References

1. Acute and chronic heart failure guidelines, https://www.escardio.org/Guidelines/Clinical-Practice-Guidelines/Acute-and-Chronic-Heart-Failure
2. Mimic iv, https://mimic-iv.mit.edu/
3. van der Aalst, W.: Process Mining. Springer Berlin Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4, https://doi.org/10.1007/978-3-662-49851-4
4. Hompes, B., Buijs, J., Aalst, W., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering (11 2015)
5. de Leoni, M., van der Aalst, W.M., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Information Systems **56**, 235–257 (Mar 2016). https://doi.org/10.1016/j.is.2015.07.003, https://doi.org/10.1016/j.is.2015.07.003
6. Levey, A.S., Perrone, R.D., Madias, N.E.: Serum creatinine and renal function. Annu Rev Med **39**, 465–490 (1988)
7. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Measuring the precision of multi-perspective process models. In: Business Process Management Workshops, pp. 113–125. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-42887-1$_1$0, https://doi.org/10.1007/978-3-319-42887-1_10
8. Martin, N., De Weerdt, J., Fernández-Llatas, C., Gal, A., Gatta, R., Ibáñez, G., Johnson, O., Mannhardt, F., Marco-Ruiz, L., Mertens, S., Munoz-Gama, J., Seoane, F., Vanthienen, J., Wynn, M.T., Boilève, D.B., Bergs, J., Joosten-Melis, M., Schretlen, S., Van Acker, B.: Recommendations for enhancing the usability and understandability of process mining in healthcare. Artificial Intelligence in Medicine **109**, 101962 (2020). https://doi.org/https://doi.org/10.1016/j.artmed.2020.101962, http://www.sciencedirect.com/science/article/pii/S0933365720312276
9. Olokoba, A.B., Obateru, O.A., Olokoba, L.B.: Type 2 diabetes mellitus: a review of current trends. Oman Med J **27**(4), 269–273 (Jul 2012)
10. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In: Lohmann, N., Song, M., Wohed, P. (eds.) Business Process Management Workshops. pp. 15–27. Springer International Publishing, Cham (2014)
11. Rozinat, A., Mans, R., Song, M., van der Aalst, W.: Discovering simulation models. Information Systems **34**(3), 305–327 (May 2009). https://doi.org/10.1016/j.is.2008.09.002, https://doi.org/10.1016/j.is.2008.09.002
12. Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: A framework for efficiently mining the organisational perspective of business processes. Decision Support Systems **89**, 87–97 (Sep 2016). https://doi.org/10.1016/j.dss.2016.06.012, https://doi.org/10.1016/j.dss.2016.06.012
13. Schönig, S., Ciccio, C.D., Maggi, F.M., Mendling, J.: Discovery of multi-perspective declarative process models. In: Service-Oriented Computing, pp. 87–103. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-46295-0$_6$, https://doi.org/10.1007/978-3-319-46295-0_6

14. Wynn, M., Poppe, E., Xu, J., ter Hofstede, A., Brown, R., Pini, A., van der Aalst, W.: Processprofiler3d: A visualisation framework for log-based process performance comparison. Decision Support Systems **100**, 93 – 108 (2017). https://doi.org/https://doi.org/10.1016/j.dss.2017.04.004, http://www.sciencedirect.com/science/article/pii/S0167923617300623, smart Business Process Management
15. Yasmin, F., Bukhsh, F., Silva, P.: Process enhancement in process mining: A literature review (12 2018)

Detecting Semantic Business Process Model Clones

Thomas S. Heinze[1], Wolfram Amme[2], and André Schäfer[2]

[1] German Aerospace Center (DLR)
thomas.heinze@dlr.de
[2] Friedrich Schiller University Jena
[wolfram.amme,andre.schaefer]@uni-jena.de

Abstract. Process modeling with languages like BPMN allows process designers to create the same business process model in various ways. Detecting model clones, i.e., pairs of business process models sharing a certain degree of similarity can be difficult. In this paper, we propose an approach to process model clone detection based upon dominator trees and targeted at detecting semantically though not necessarily syntactically similar process models of business processes.

1 Introduction

Duplicated process models is a common issue in business process management and modeling and in particular relevant when process models are organized in model repositories [11]. Matching business process models and estimating their similarity has thus been an important research topic and has many applications, ranging from analyzing conformance to reference models [3], tracing and identification of process variants [2] to process model search and clone detection [1]. In this paper, we address the latter problem of finding process models which share a certain degree of similarity, which is known as model clone detection in the literature [11,12]. Note that such model clones can origin from homologous development [14], where a process designer reuses and modifies an existing model to generate a new process model. As a result, the original and new model usually share a high lexical similarity. In contrast, in heterologous development [14], process models are created independently of each other but implementing similar functionalities. Thus, such models are not necessarily syntactically similar, i.e., can differ in the number of activities and gateways or in their structure. Finding such semantic clones is in particular hard for business process models due to the large number of modeling purposes and practices, e.g., block- and graph-oriented process modeling styles. While differing modeling styles can be found in different business process modeling languages, multiple paradigms can even exist in the very same language. A well-known representative for such a language is BPMN.

As an example, consider the two BPMN process models shown on the left-hand side of Fig. 1. Apparently, both process models implement a similar business process. In particular, in terms of possible execution traces, the lower model's

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

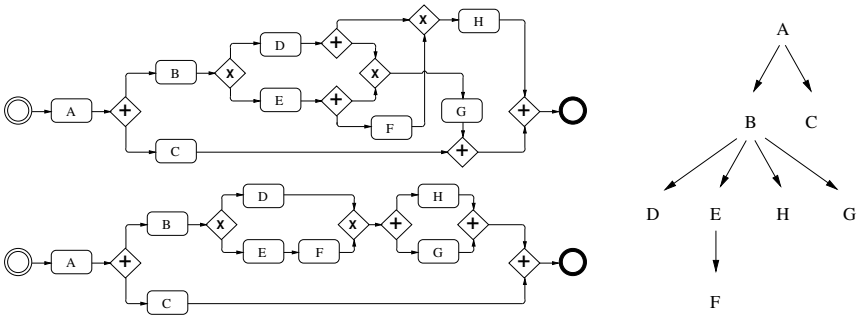


Fig. 1. Business process models (left) and dominator tree (right)

behavior is a subset of the upper model's behavior (activity G can precede activity F in the upper model in contrast to the lower model). Estimating the models' similarity and therefore identifying them as model clones is though not straightforward due to the more graph-oriented modeling style of the upper model and the more block-oriented style of the lower model. Such a pair of process models is known as a semantic model clone, i.e., two process models which have similar behavior but do not necessarily share the same syntax and structure [12].

2 Process Model Clone Detection

In order to identify semantic model clones, we propose a model clone detection method based on dominator trees [10]. Dominator trees provide for an abstraction of process control flow and therefore encode the dominance relation of flow graphs. A node n in a flow graph is said to dominate another node m, iff n comes before m on every path from the flow graph's start node to m. Note that this relation can be efficiently computed for flow graphs using data flow analysis, also for workflow graphs [4,8]. Each node and its immediate, i.e., closest dominating node then results in an edge in the dominator tree. The same dominator tree represents both the example models and is shown on the right-hand side of Fig. 1. As can be seen, the dominator tree comprises guaranteed happens-before relations between the models' activities, while abstracting from the process models' structure.

In order to allow for an efficient comparison of process models, dominator trees are encoded into sets of integer sequences. In this way, standard metrics, e.g., Hamming or edit distance, can be used for their comparison. If the thus computed difference between two process models does not exceed a certain threshold value, the models are considered to be a model clone. For the encoding, each path of a dominator tree, starting at its root node and ending at a leaf node, is mapped to a sequence of integers, yielding a set of integer sequences. For the dominator tree in Fig. 1, this would mean to encode the five paths $[A, B, D]$, $[A, B, E, F]$, $[A, B, H]$, $[A, B, G]$, and $[A, C]$. As the two example process models share the same dominator tree shown in Fig. 1, they are identified as model clone. Also

considering modifications like adding, modifying, or removing a single or a small number of nodes in one of the models, as in case of homologous development, would only slightly affect their encoding and the result of their comparison.

Control flow is an integral part of a business process model, but its other aspects have to be considered as well. While the encoding of dominators trees introduced above does not cover the analysis of node labels by itself, stemming, bag-of-words, word embeddings, and other related methods [6,11] can be integrated into our approach. In this way, the encoding becomes more permissive in the presence of differing node labels, as is common in business process models. Furthermore, the pre-processing of node labels by means of Static Single Assignment Form [4,8] helps in including aspects of process data in the encoding [4]. The thorough study of an optimal encoding is subject to our future work.

3 Related Work

Business process model similarity estimation and matching has been a frequent topic in research. Due to space constraints, we refer to the survey in [11] for a comprehensive overview. Respective approaches can be categorized into: syntactical, structural, behavioral methods and approaches based upon human judgement. Syntactical methods compare process models based on the therein used labels, e.g., as in [6]. Structural methods use process models as graphs for estimating their similarity, e.g., calculating the graph edit distance or isomorphisms [1]. Behavioral methods instead use process execution traces or logs to compare process models, e.g., measuring the longest common subseqence [3]. Note that our proposed method aligns between both, as it does not rely on the actual process modeling structure. In this way, we avoid the usually costly analysis of process behavior but are still able to abstract from modeling styles and practices. The comparison with behavioral methods and analysis of the tradeoff between precision and scalability to large process model repositories is subject to future work. In a way, our method reminds of causal footprints [2] or behavioral profiles and footprints [7,13]. Remember that dominator trees encode the guaranteed happens-before relation of activities. Behavioral profiles and footprints are based on execution traces and the direct or eventual successor relation of activities therein, respectively. Measuring similarity is then conducted using the relations' Jaccard index for two process models, or an execution log and a process model. In general, the major part of research on clone detection addresses source code. However, some approaches have been considering clone detection for model-based languages besides business process modeling languages, e.g., UML [12].

4 Next Steps

We are interested in implementing the proposed method in a system for detecting semantic BPMN model clones, e.g., starting with the existing control flow analyses available in mojo [9]. The implementation would allow us for experimenting with various optimizations and parameters, including differing metrics and encodings.

Furthermore, the thorough evaluation and comparison with state-of-the-art approaches to process model clone detection and similarity estimation, in particular the trace-based approaches mentioned above, is another item for future work. Such an evaluation requires though a dataset with ground truth, i.e., known process model clones. As such datasets are unfortunately scarce, we plan to resort to human judges for generating missing labels in an existing dataset, using for example the set of mined process models in [5]. Alternatively, applying small mutations on seed models can be used to generate a dataset, similar to [7].

References

1. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011)
2. van Dongen, B., Dijkman, R., Mendling, J.: Measuring Similarity between Business Process Models. In: CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer (2008)
3. Gerke, K., Cardoso, J.S., Claus, A.: Measuring the Compliance of Processes with Reference Models. In: OTM 2009. LNCS, vol. 5870, pp. 76–93. Springer (2009)
4. Heinze, T.S., Amme, W., Moser, S.: Static analysis and process model transformation for an advanced business process to Petri net mapping. Softw. Pract. Exp. **48**(1), 161–195 (2018)
5. Heinze, T.S., Stefanko, V., Amme, W.: Mining BPMN Processes on GitHub for Tool Validation and Development. In: BPMDS/EMMSAD@CAiSE 2020. LNBIP, vol. 387, pp. 193–208. Springer (2020)
6. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing Recall of Process Model Matching by Improved Activity Label Matching. In: BPM 2013. LNCS, vol. 8094, pp. 211–218. Springer (2013)
7. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity – A Proper Metric. In: BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer (2011)
8. Lee, J., Midkiff, S.P., Padua, D.A.: Concurrent Static Single Assignment Form and Constant Propagation for Explicitly Parallel Programs. In: LCPC 1997. LNCS, vol. 1366, pp. 114–130. Springer (1997)
9. Prinz, T.M., Spieß, N., Amme, W.: A First Step towards a Compiler for Business Processes. In: CC 2014. LNCS, vol. 8409, pp. 238–243. Springer (2014)
10. Schäfer, A., Amme, W., Heinze, T.S.: Detection of Similar Functions Through the Use of Dominator Information. In: ACSOS 2020 Comp. pp. 206–211. IEEE (2020)
11. Schoknecht, A., Thaler, T., Fettke, P., Oberweis, A., Laue, R.: Similarity of Business Process Models – A State-of-the-Art Analysis. ACM Comput. Surv. **50**(4), 52:1–52:33 (2017)
12. Störrle, H.: Towards Clone Detection in UML Domain Models. Softw. Syst. Model. **12**(2), 307–329 (2013)
13. Weidlich, M., van der Werf, J.M.: On Profiles and Footprints – Relational Semantics for Petri Nets. In: Petri Nets 2012. LNCS, vol. 7347, pp. 148–167. Springer (2012)
14. Wu, M., Wang, P., Yin, K., Cheng, H., Xu, Y., Roy, C.K.: LVMapper: A Large-Variance Clone Detector Using Sequencing Alignment Approach. IEEE Access **8**, 27986–27997 (2020)

A Dashboard-based Approach for Monitoring Object-Aware Processes

Marius Breitmayer, Lisa Arnold and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{marius.breitmayer,lisa.arnold,manfred.reichert}@uni-ulm.de

Abstract. Data (e.g., event logs) gathered during the execution of business processes enable valuable insights into actual process performance. To leverage this knowledge, these data should be analyzed and interpreted in the context of the respective processes. Corresponding analyses allow for a comprehensive process monitoring as well as the discovery of weaknesses and potential process improvements. This also applies to object-aware processes, where data drives process execution and, thus, is treated as first-class citizen. This paper introduces a dashboard with advanced monitoring functions for object-aware processes.

Keywords: object-aware processes, process monitoring, dashboard

1 Introduction

Process monitoring leverages the data created during the execution of business processes in order to gain insights into actual process performance and to ensure process conformance with regulations, policies, and business rules [12]. This information, in turn, can then be used to monitor, analyze and improve processes [8]. Despite existing approaches for monitoring activity-centric processes (e.g., [5]), adequate monitoring support for data-centric paradigms to business process management (BPM) [16], e.g., artifact-centric processes [7], object-aware processes [10], and case handling [2], is still lacking. As opposed to activity-centric processes, in object-aware process management, large process structures involving multiple concurrently executed and interacting object lifecycle processes, emerge during runtime [15,4]. The monitoring of such dynamically evolving process structures constitutes a particular challenge not sufficiently addressed by contemporary approaches [14]. As data is treated as first-class citizen in object-aware process management, however, the monitoring of data-centric processes yields great potential for more advanced analyses and better comprehensibility of processes in general due to the tight integration of process and data.

To set a background, a short introduction of object-aware process management is provided. PHILharmonicFlows, our approach to data-centric BPM [10], introduces the concepts of *objects*, *object behavior,* and *object interaction*. Each business object of a real-world business process is represented as one such object. The latter comprises data, represented in terms of *attributes,* and a state-based

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

process model describing object behavior in terms of an *object lifecycle* model. In a practical application of the PHILharmonicFlows system, which resulted in a data- and process-aware e-learning system, examples of business objects include *Exercise*, *Submission,* and *Lecture* [3]. At runtime, when necessary data (i.e., object attributes), such as *Points* or *Feedback,* become available, this enables the transition between the states of a lifecycle. Finally, interactions between object lifecycles are managed by coordination processes [13].

Section 2 discusses related work. Section 3 sketches three approaches for calculating process progress, followed by a description of the monitoring dashboard in Section 4. Finally, Section 5 provides a summary and outlook.

2 Related Work

In the BPM lifecycle, process monitoring is a key component for analyzing and improving processes [8,17]. A wide range of approaches and tools exist for activity-centric processes, especially in the field of process mining [1]. To identify problems of a process, conformance checking verifies whether a process is executed as intended by replaying or aligning an event log with the corresponding process model [6]. However, there only exist few approaches for monitoring data-centric processes, which focus on artifact-based processes [11]. Process monitoring and conformance checking for activity-centric processes mainly focus on the identification of bottlenecks, deviations between event log and process model, and the prediction of the next activities [9]. By contrast, our work deals with the monitoring of object-aware processes and the large process structures emerging in this context during runtime.

3 Towards Progress Calculation

To understand the calculation of progress metrics displayed in the dashboard (c.f. Fig. 1), this section describes the three approaches implemented for calculating the progress of object lifecycle instances. Though none of the described calculations covers all aspects, their combination (e.g., individually weighted average) constitutes a good approximation and is therefore supported by the dashboard.

Approach 1 (Lifecycles). This approach divides the number of completed lifecycle steps by the number of total lifecycle steps for each object instance. Note that this calculation is non-trivial as different steps may be executed depending on data (and routing decisions). Therefore, the number of completed steps may vary depending on the path taken through the lifecycle. To reconstruct the latter, for each lifecycle step it is checked whether a value is assigned in the event log. This allows determining the actual number of steps executed for each object instance, which then may be divided by the number of remaining steps.

Approach 2 (Relations). Using the hierarchies specified in the data model, the progress of an object instance can be calculated as the average progress of its lower-level instances. This approximates how constraints between states of object instances influence the progress of the overall object-aware process.

Approach 3 (Temporal Distance). This approach calculates the temporal distance between object instantiation and a certain point in time in the event log. To calculate the progress, this distance may then be divided by the average duration of all completed object instances of the same type.

4 Monitoring Dashboard

The monitoring of an object-aware process needs to consider the information contained in its *data model* (e.g., objects, attributes, relations, hierarchies), its object *lifecycles* (e.g., steps and states), and its *coordination process* [10] in conjunction with the data recorded during process execution (e.g., event logs). We created a monitoring dashboard that allows us to display various aspects of an object-aware process (see Fig. 1). As input, this dashboard takes an object-aware process (i.e., the data model, lifecycles and coordination process) and the event log created during process enactment. Note that event logs of object-aware processes differ from the ones of activity-centric processes. While the latter consists of case-related activity events enriched with additional information [1], a data-centric event log contains information about object instances (e.g., ids, object types, attributes and their values), lifecycle processes (e.g., steps, states, state changes), and object interactions. Using this information, the monitoring dashboard can provide an overview of the overall object-aware process (see Fig. 1), as well as drill-down and roll-up functions for inspecting selected aspects. The dashboard was evaluated for two processes with corresponding event logs.

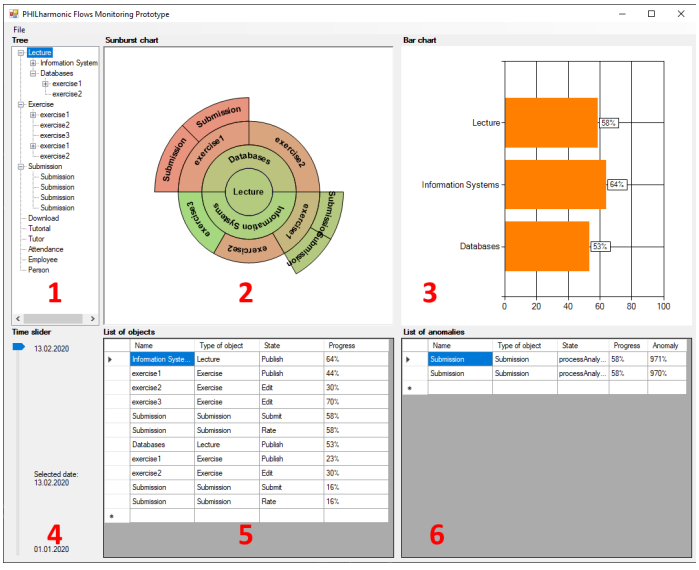


Fig. 1. PHILharmonicFlows Monitoring Dashboard

1. The *tree navigation element* lists the various object types (e.g., *Lecture*, *Exercise*, *Submission*) derived from the data model, together with the corresponding instances (e.g., "Databases" and "Information Systems" as instances of object *Lecture*) recorded in the event log. Additionally, the tree navigation displays lower-level instances of any object instance that can be derived from the hierarchical structuring of the objects in the data model (e.g., "Exercise1" as lower-level instance of the "Database" lecture).
2. The *sunburst chart* provides an overview of selected objects, together with their instances and corresponding lower-level instances. Each layer contains the object instances belonging to the same hierarchical level (e.g., all exercises of a lecture series). However, when the number of object instances grows, the ratio of each individual instance shrinks. To tackle this issue, we implemented "drill-down" and "roll-up" functions. This allows inspecting individual process aspects (e.g., object instances) instead of the entire process. Additionally, color coding of elements enables fast bottleneck identification.
3. The *bar chart* provides a quick and easy method to either compare the progress of similar object instances (see Fig. 1) or the average progress of instances of the same type. We added "drill-down" and "roll-up" functions.
4. The *time slider* allows displaying the state of the object-aware process at any point in time based on recorded event logs. If a point in time other than the most recent one is selected, the status of displayed instances is reconstructed through partial event log replay. As a result, all dashboard elements display the state of the process at the selected point in time. This allows replaying the process as well as detecting former issues that have been resolved.
5. The *list of object instances* displays additional information of the object instances being of interest. The table may be sorted according to any criterion.
6. The *anomalies table* lists object instances that are likely to be outliers. To identify these outliers, the dashboard allows for the comparison with a relative or absolute threshold. For relative comparison, this is accomplished using the following formula:
   $(1 + \frac{threshold(\%)}{100}) < \frac{InstanceValue}{ObjectAverage}$ or $\frac{InstanceValue}{ObjectAverage} < (1 - \frac{threshold(\%)}{100})$
   For absolute comparison, the difference between an object instance and the object median is calculated and compared with the absolute threshold (e.g., 3 days). To enable a more sophisticated outlier detection, the dashboard is able to combine both methods using AND/OR operations.

5 Conclusion and Outlook

This paper presents a monitoring dashboard for object-aware processes which enables advanced monitoring functions. It combines knowledge from the object-aware process model with information about the execution to visualize the states of all object instances at any moment during the process execution. This allows detecting bottlenecks and outliers. Additionally, the dashboard was tested with two real-world processes and corresponding event logs. In future work, we will improve outlier detection based on more sophisticated algorithms and incorporate different user perspectives (i.e., personalized monitoring views).

Acknowledgments This work is part of the ZAFH Intralogistik, funded by the European Regional Development Fund and the Ministry of Science, Research and Arts of Baden-Württemberg, Germany (F.No. 32-7545.24-17/3/1)

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer (2016)
2. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. DKE **53**(2), 129–162 (2005)
3. Andrews, K., Steinau, S., Reichert, M.: Engineering a highly scalable object-aware process management engine using distributed microservices. In: Int'l Conf on CoopIS'18. pp. 80–97 (2018)
4. Andrews, K., Steinau, S., Reichert, M.: Enabling runtime flexibility in data-centric and data-driven process execution engines. Information Systems p. 101447 (2019)
5. Bülow, S., Backmann, M., Herzberg, N., Hille, T., Meyer, A., Ulm, B., Wong, T.Y., Weske, M.: Monitoring of business processes with complex event processing. In: Business Process Management Workshops. Springer (2014)
6. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer (2018)
7. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
8. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, 2nd edn. (2018)
9. Jorbina, K., et al: Nirdizati: A web-based tool for predictive process monitoring. In: BPM Demo Track and BPM Dissertation Award (CEUR Workshop Proceedings, Volume 1920), pp. 1–5 (2017)
10. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. J of Soft Maint & Evo **23**(4), 205–244 (2011)
11. Meroni, G.: Artifact-driven Business Process Monitoring. Ph.D. thesis, Politecnico di Milano Milan Italy (2018)
12. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Berlin-Heidelberg (2012)
13. Steinau, S., Andrews, K., Reichert, M.: Modeling process interactions with coordination processes. In: CoopIS'18. pp. 21–39. LNCS, Springer (2018)
14. Steinau, S., Andrews, K., Reichert, M.: The relational process structure. In: CAiSE 2018. pp. 53–67. No. 10816 in LNCS, Springer (2018)
15. Steinau, S., Andrews, K., Reichert, M.: Executing lifecycle processes in object-aware process management. In: Data-Driven Process Discovery and Analysis. pp. 25–44. Springer (2019)
16. Steinau, S., Marrella, A., Andrews, K., Leotta, F., Mecella, M., Reichert, M.: DALEC: A framework for the systematic evaluation of data-centric approaches to process management software. Softw & Sys Modeling **18**(4), 2679–2716 (2019)
17. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2019)

Custom-MADE – Leveraging Agile Rationale Management by Employing Domain-Specific Languages

Mathias Schubanz

Brandenburg University of Technology,
Platz der Deutschen Einheit 1, 03046 Cottbus, Germany
M.Schubanz@b-tu.de

Abstract. Managing rationale in software development projects can be a cumbersome task with a potentially low return on investment. Especially in the agile context, documentation is therefore very unpopular. Research has not yet properly addressed an agile documentation workflow. In this paper, the author presents an integrated approach to agile rationale management based on a highly-flexible modelling approach using domain-specific languages. It facilitates the complete documentation workflow from capture to reuse, partially automates it and offers various customisation opportunities, making it applicable to agile methods.

Keywords: agile · decision-making · Language Server Protocol · domain-specific languages · documentation · tool support · rationale management

1 Introduction

Managing rationale can considerably improve comprehension in software development. It facilitates change impact analysis as well as requirements traceability and evolution [35]. Moreover, it improves the understanding of architectural decisions and leads to better decisions [36]. In *agile software development* (ASD) this can be particularly important, as empirical work showed that stakeholders in agile teams perceive less architecture involvement [16]. Furthermore, in ASD documentation is questioned with particular scepticism. ASD instead promotes working software while depreciating documentation (cf. *Agile Manifest* [3]). Despite the mentioned and other potential benefits (cf. Tang et al. [34]), the structured and systematic handling of decisions is only applied seldom in practise [2,11]. Even in ASD teaching, it is only dealt with very selectively [22].

As part of an ongoing research project [32] with the focus adapted to ASD, the author developed *Custom-MAnangement of DEcision* (Custom-MADE), an integrated process-centric approach to rationale management. Based on domain-specific languages (DSL) it integrates a highly-flexible modelling approach enabling its users to tailor it to individual or enterprise-wide needs and documentation standards. Another feature is its minimal-invasive and generic approach offering to combine it with existing workflows. All these customization opportunities are vital for use in individually tailored processes in ASD.

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

The remainder of the paper starts with a presentation of related work (Section 2). Section 3 presents a simplified rationale management workflow, while Section 4 elaborates on how *Custom-MADE* enhances it. Subsequently, Section 5 introduces the model architecture and Section 6 the chosen tool architecture. Finally, Section 7 concludes the paper and provides an outlook on future research.

2 Related Work

Approaches to direct process integration of rationale management in ASD are sparse, if at all. Rather, there are many tool-based approaches for rationale management in ASD. In the field of agile requirements gathering, for instance, Lee et al. [24] developed *Echo*, an approach connecting requirements and related design decisions. Around the same time Sauer [30] presented an approach for automating the capture of rationale in ASD. It aims to reduce costs by linking historical data and prototype definitions to an event-based rationale model. More recently, Hadar et al. [15] introduced an approach to document an architecture in the elevator speech concept. Their tool helps architects in organizing relevant information while creating design and architecture blueprints, thus reducing documentation effort. Also recently Voigt et al. [37] presented *sprintDoc*, a tool-based concept based on *DokuWiki* [13]. It integrates the documentation of artefacts into the agile process and thus traces changes in documents along with changes in issues.

There are also a lot of contributions around rationale modelling. These contributions often include tools that support rationale capture or sometimes even the decision-making process, as e. g., Miksovic and Zimmerman [28]. Some approaches even start with requirements analysis, such as RADAR [12]. A considerable share of the model-based contributions stems from research on software architecture documentation, less from the field of agile documentation. Since *Custom-MADE* aims less at architecture-bound documentation and more at integration into lightweight agile processes, please refer to Tang et al. [33] for a comprehensive overview of other approaches to modelling software architecture decisions as well as suitable tool support.

When considering the modeling approach from a more general perspective, one of the approaches that is most similar to *Custom-MADE* is *Frag* [38]. However, Zdun focuses more on DSL-based designs rather than using DSLs to get maximum flexibility for the documentation models.

Further work tries to leverage the opportunity of capturing rationale exactly where and when they are made to mitigate a substantial barrier to rationale capture (cf. [11]). For instance, *DesignMinders* [8] complements whiteboard systems so that rationale are directly digitised and browseable. Other approaches directly integrate with the IDE, as implemented by *SEURAT* [10] and *DecDoc* [17]. *ConDec* [21] even goes one step further by additionally integrating with *JIRA* [25] and *Slack* [26].

Other related contributions ignore formal and process aspects of rationale documentation entirely. They focus on retrospectively extracting rationale from existing documents by, e. g., integrating machine learning techniques (cf. Alkadi et al. [1], Bhat et al. [6], or Rogers et al. [29]).

3 Rationale Management Workflow

As presented in the related work (cf. Section 2), many of the approaches to date focus on either modelling, the activity of capturing, or persisting and providing design rationales to developers. However, only a few approaches focus on a holistic workflow and a possible process integration with it. In this chapter, the author presents a simplified rationale management workflow that serves as the basis for presenting the tool-driven approach later on (cf. Figure 1). As of now, *Custom-MADE* deliberately ignores the reasoning and decision-making process and begins with the rationale capture.

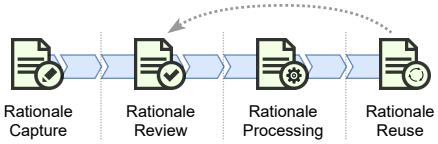


Fig. 1. Rationale management workflow.

Initially, (*I*) rationale that are considered sufficiently important will be documented. As a second step, (*II*) the rationale are reviewed at a given time (e.g. in retrospective, as in [31]). Following the review and potential modifications, (*III*) the documented rationale are processed and archived for later use. The last activity (*IV*) constitutes the reuse of the recorded information. If it is necessary to modify already documented rationale, the workflow can be reiterated from the review.

4 Enhanced Rationale Management Workflow

The aim of *Custom-MADE* is to enable an individual, customisable, flexible and semi-automated rationale management workflow. *Custom-MADE* starts even before the actual workflow (cf. Figure 2).

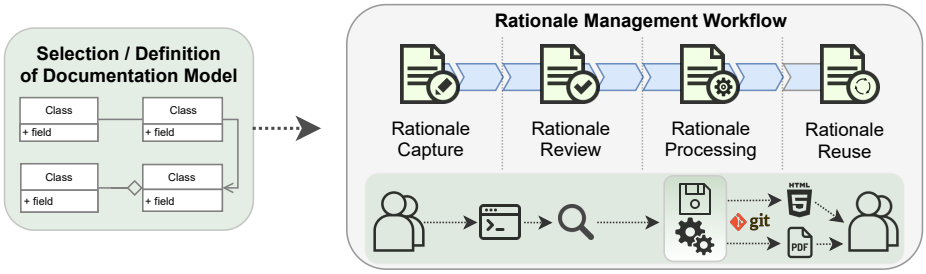


Fig. 2. Custom-MADE – Enhanced rationale management workflow.

With domain experts, developers should either define a particular documentation model or select one from a set of predefined models. Building on the

underlying DSL technology, *Custom-MADE* users can define their particular models and integrate them seamlessly into the toolchain. In the following, the author describes modifications to the workflow and how the tool facilitates the rationale management workflow:

Rationale Capture / Rationale Review

To support software engineers during the *Rationale Capture*, *Custom-MADE* provides an easy and intuitive web-based editor facilitating the previously defined documentation model (cf. Figure 4). Based on the *Xtext* framework [14], this editor offers high levels of flexibility and customisability. It provides a full language infrastructure to the user, i. e., among others, auto-completion, semantic colouring, error checking, quick-fix suggestions. These and many more are individually adapted to the chosen documentation model (defined as DSL). This feature set simultaneously facilitates a semi-automated *Rationale Review* (cf. Figure 2). By offering documentation guidance in the form of customisable rationale capture templates and formal and semantic checks, the developers can fully focus on the content aspects when reviewing the documented design rationale. *Custom-MADE* takes over the remaining part of the quality control in a semi-automated way.

Rationale Processing

Complying with enterprise-wide documentation standards and making documentation available to others can be the most cumbersome documentation task. Here *Custom-MADE* steps in and triggers a processing pipeline when saving the documented rationale. For each documentation model, there are either predefined or specifically customisable generators that transfer the documented rationale into the desired and easily accesible formats, as, e. g., *Markdown Architecture Decision Records* [23], *HTML* [4], or *PDF* [7]. The generators can be easily adapted using *Xtend* [5] to make the documentation comply with existing standards and generate desired formats.

Rationale Reuse

As mentioned already, it is often cumbersome to use documentation that has already been produced. Accessibility and availability are central obstacles to effective usage here. *Custom-MADE* addresses these by storing the raw documentation and the generated files on the software developers' central workplace, the code repository. The user has the opportunity to connect a project to a *git* repository (cf. [27]) on a remote git server. If connected, the documentation will be stored and versioned on a separate, individual development branch. This storage concept not only enables developers to access older versions but also centralises the storage location. Thus, it is always clear where the documentation can be found. The storage concept also enables the direct use of documentation, e.g. in Markdown format, within the IDE. It also mitigates accessibility barriers by enabling additional services, such as full-text search usually offered by IDEs. Corresponding search functions for the web interface are also being developed.

5 Model Architecture

For the implementation of the desired modelling flexibility, it was necessary to choose a dynamically configurable approach. Accordingly, the author decided to implement a model concept with three levels (cf. Figure 3).

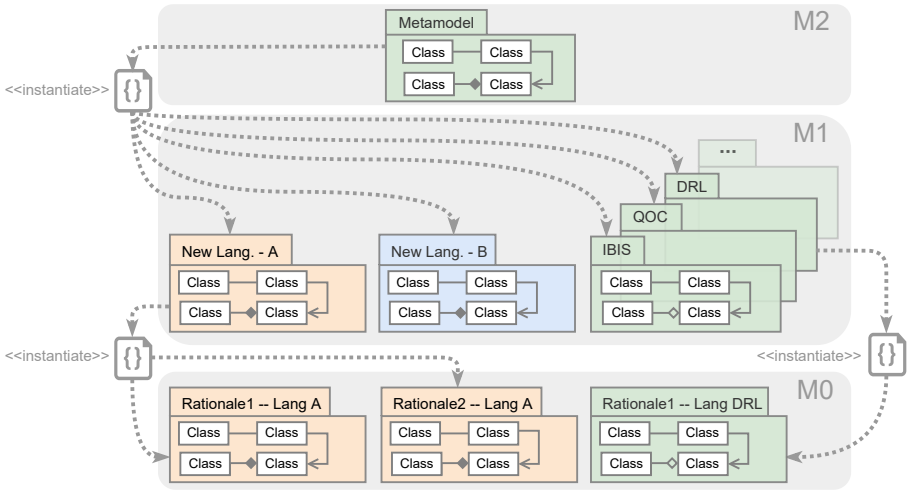


Fig. 3. Language modelling architecture.

On the top level (cf. level M2, Figure 3), a meta-model is used, which can implement the common modelling approaches from rationale management (cf. green packages). This meta-model is now interpreted as a DSL grammar. In this way, a language server can be generated that provides a complete language infrastructure for the definition at the model level (cf. level M1, Figure 3). This way, users can create their documentation model (cf. orange / blue package on the M1 level) with the *Monaco Editor* (cf. Figure 4). Once this is complete, the model itself is interpreted as a DSL grammar to generate the language infrastructure for the rationale documentation (cf. level M0). Starting from this point, developers can now capture their rationale and start the rationale management workflow.

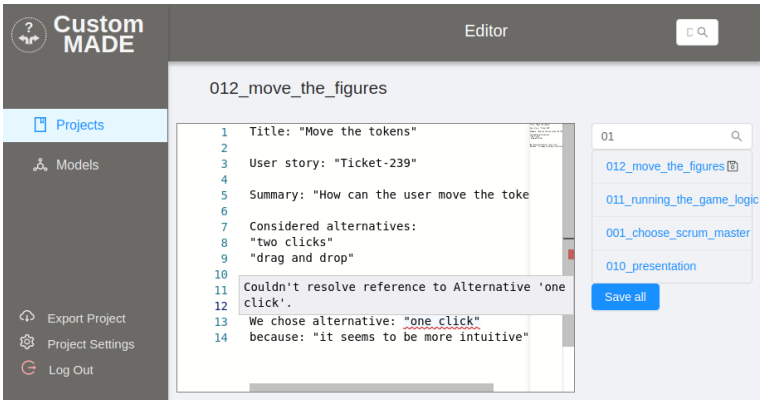


Fig. 4. Screenshot of the integrated editor showing semantic and syntactic hints.

6 Custom-MADE Tool Architecture

The author briefly presents the tool architecture based on the overview illustration in Figure 5. The left-hand side shows the part of *Custom-MADE* that is visible to the user. Based on a *ReactJS* web application, a *spring*-backend and a special implementation of the *Monaco Editor*, all functionalities described in Section 4 are accessible to the user. Particular attention is to be paid to the *Monaco Editor*, as its support of the *Language Server Protocol* (LSP) (cf. [9]) enables the flexibility in modelling that is one of the core features of the *Custom-MADE* approach. With the help of this web interface, developers can initially select or individually define their documentation model. Using *Xtext*, an individual language server is generated, with which the *Monaco Editor* supports the complete language infrastructure. The server and the editor communicate via JSON-RPC [20]. The user can now start to record design rationales based on the defined documentation model. These are validated instantly in the editor and in the processing step (cf. centre of Figure 5). If required, test cases can also be implemented for the model. Subsequently, with the generators' help the defined document formats are created and stored in the project's code repository (cf. right side of Figure 5).

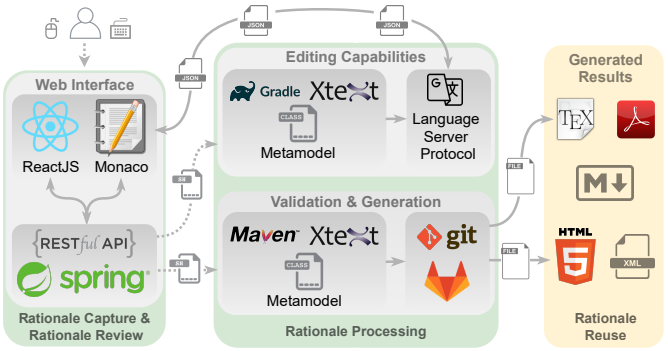


Fig. 5. Abstract tool architecture.

7 Conclusion and Outlook

In this paper, the author presented a process-centric approach for the partial automation of rationale documentation, called *Custom-MADE*. Special features include its model-flexibility and customisability at various points in the toolchain. It can be easily applied to existing workflows and configured with documentation models already in use.

Future work includes, but is not limited to, the implementation of profound traceability down to the ticket level and automatically generated review requests becoming necessary due to changes that affect other rationale documentation.

Acknowledgments

Special gratitude goes to Jost-V. Schulz and Sebastian Brüggemann for contributions to *Custom-MADE* and to Claus Lewerentz for his valuable feedback.

References

1. Alkadhi, R., Laţa, T., Guzman, E., Bruegge, B.: Rationale in Development Chat Messages: an Exploratory Study. In: Proceedings of the 14^{th} International Conference on Mining Software Repositories. pp. 436–446. IEEE Press (2017)
2. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edn. (2003)
3. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunnigham, W., Fowler, M., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development. http://agilemanifesto.org/ (February 2001), http://agilemanifesto.org/
4. Berners-Lee, T., Connolly, D.: Hypertext markup language-2.0 (1995)
5. Bettini, L.: Implementing Domain-Specific Languages With Xtext and Xtend. Packt Publishing Ltd (2016)
6. Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., Matthes, F.: Automatic Extraction of Design Decisions From Issue Management Systems: A Machine Learning Based Approach. In: European Conference on Software Architecture. pp. 138–154. Springer (2017)
7. Bienz, T., Cohn, R., Adobe Systems (Mountain View, C.: Portable document format reference manual. Citeseer (1993)
8. Bortis, G.: Informal Software Design Knowledge Reuse. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering. vol. 2, pp. 385–388. IEEE (2010)
9. Bünder, H.: Decoupling Language and Editor – The Impact of the Language Server Protocol on Textual Domain-Specific Languages. In: Proceedings of the 7^{th} International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2019). pp. 131–142 (2019)
10. Burge, J., Brown, D.: SEURAT: Integrated Rationale Management. In: 2008 ACM/IEEE 30th International Conference on Software Engineering. pp. 835–838. IEEE (2008)
11. Burge, J.E., Brown, D.C.: Software Engineering Using RATionale. Journal of Systems and Software **81**(3), 395–413 (2008)
12. Busari, S.A., Letier, E.: RADAR: A Lightweight Tool for Requirements and Architecture Decision Analysis. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). pp. 552–562. IEEE (2017)
13. DokuWiki: DokuWiki – It's Better When its Simple (Jul 2004), https://www.dokuwiki.org/
14. Eysholdt, M., Behrens, H.: Xtext: Implement Your Language Faster Than the Quick and Dirty Way. In: Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. pp. 307–309 (2010)
15. Hadar, I., Sherman, S., Hadar, E., Harrison, J.J.: Less is More: Architecture Documentation for Agile Development. In: 6^{th} International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). pp. 121–124. IEEE (2013)
16. Heijenk, F., van den Berg, M., Leopold, H., van Vliet, H., Slot, R.: Empirical Insights Into the Evolving Role of Architects in Decision-Making in an Agile Context. In: European Conference on Software Architecture. pp. 247–264. Springer (2018)
17. Hesse, T.M., Kuehlwein, A., Roehm, T.: DecDoc: A Tool for Documenting Design Decisions Collaboratively and Incrementally. In: 1^{st} International Workshop on Decision Making in Software ARCHitecture (MARCH). pp. 30–37. IEEE (2016)

18. Jansen, A., Van Der Ven, J., Avgeriou, P., Hammer, D.K.: Tool Support For Architectural Decisions. In: Software Architecture, 2007. WICSA'07. The Working IEEE/IFIP Conference on. pp. 4–4. Ieee (2007)
19. Jansen, A., Bosch, J.: Software Architecture as a set of Architectural Design Decisions. In: Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on. pp. 109–120. IEEE (2005)
20. JSON-RPC Working Group, .: JSON-RPC 2.0 Specification (2013)
21. Kleebaum, A., Johanssen, J.O., Paech, B., Bruegge, B.: Sharing and Exploiting Requirement Decisions. In: In Proceedings: Fachgruppentreffen Requirements Engineering (FGRE'19) (2019)
22. Kleebaum, A., Johanssen, J.O., Paech, B., Bruegge, B.: Teaching Rationale Management in Agile Project Courses. In: Tagungsband des 16. Workshops "Software Engineering im Unterricht der Hochschulen" (2019)
23. Kopp, O., Armbruster, A., Zimmermann, O.: Markdown Architectural Decision Records: Format and Tool Support. In: ZEUS. pp. 55–62 (2018)
24. Lee, C., Guadagno, L., Jia, X.: An Agile Approach to Capturing Requirements and Traceability. In: Proceedings of the 2^{nd} International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). vol. 20 (2003)
25. Li, P.: Jira 7 Essentials. Packt Publishing Ltd (2016)
26. Lin, B., Zagalsky, A., Storey, M.A., Serebrenik, A.: Why developers are slacking off: Understanding how software teams use slack. In: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion. pp. 333–336 (2016)
27. Loeliger, J.: Version control with Git. O'Reilly Series, O'Reilly (2009), http://books.google.de/books?id=e9FsGUHjR5sC
28. Miksovic, C., Zimmermann, O.: Architecturally Significant Requirements, Reference Architecture, and Metamodel for Knowledge Management in Information Technology Services. In: 9^{th} Working IEEE/IFIP Conference on Software Architecture. pp. 270–279. IEEE (2011)
29. Rogers, B., Qiao, Y., Gung, J., Mathur, T., Burge, J.E.: Using Text Mining Techniques to Extract Rationale From Existing Documentation. In: Design Computing and Cognition'14, pp. 457–474. Springer (2014)
30. Sauer, T.: Using Design Rationales for Agile Documentation. In: Proceedings of the 12^{th} IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE). pp. 326–331. IEEE (2003)
31. Schubanz, M., Lewerentz, C.: What Matters to Students – A Rationale Management Case Studyin Agile Software Development. In: Tagungsband des 17. Workshops "Software Engineering im Unterricht der Hochschulen", Innsbruck, Österreich (2020)
32. Schubanz, M.: Design Rationale Capture in Software Architecture: What has to be Captured? In: Proceedings of the 19^{th} International Doctoral Symposium on Components and Architecture. pp. 31–36. ACM (2014)
33. Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Babar, M.A.: A Comparative Study of Architecture Knowledge Management Tools. Journal of Systems and Software **83**(3), 352–370 (2010)
34. Tang, A., Babar, M.A., Gorton, I., Han, J.: A Survey of Architecture Design Rationale. Journal of Systems and Software **79**(12), 1792–1804 (2006)
35. Thurimella, A., Schubanz, M., Pleuss, A., Botterweck, G.: Guidelines for Managing Requirements Rationales. Software, IEEE **34**(1), 82 – 90 (2017). https://doi.org/10.1109/MS.2015.157

36. Tofan, D., Galster, M., Avgeriou, P.: Difficulty of Architectural Decisions – A] Survey With Professional Architects. In: European Conference on Software Architecture. pp. 192–199. Springer (2013)
37. Voigt, S., Hüttemann, D., Gohr, A.: SprintDoc: Concept for an Agile Documentation Tool. In: 11^{th} Iberian Conference on Information Systems and Technologies (CISTI). pp. 1–6. IEEE (2016)
38. Zdun, U.: A DSL Toolkit for Deferring Architectural Decisions in DSL-Based Software Design. Information and Software Technology **52**(7), 733–748 (2010)

ElogQP: An Event log Quality Pointer

Tobias Ziolkowski[1], Lennart Brandt[2], Agnes Koschmider[1]

[1] Process Analytics Group,
Computer Science Department, Kiel University, Germany
{tzi|ak}@informatik.uni-kiel.de
[2] stu113969@mail.uni-kiel.de

Abstract. This paper presents *ElogQP*, a tool to detect data quality violations in an event log. Data quality issues significantly impact the process discovery result. Thus, *ElogQP* represents an essential step towards improved process discovery.

Keywords: event log, process mining, data cleaning, imperfection patterns.

1 Introduction

Event log files are used as input to any process mining algorithm aiming to discover an as-is process model or to identify bottlenecks. Although recently process mining has gained an impressive uptake, still, data quality violations often hamper the direct applicability of process mining techniques on an event log. There are several reasons for data quality violations like those that the recorded event data is not saved in the correct order, data entries are missing (e.g. timestamps or case ID) or are not recorded correctly (e.g. incomplete activity names). These quality violations lead to inappropriate event logs and finally significantly impact the process discovery result. To counteract data quality issues in process mining several approaches exist [1, 2, 3] like to define maturity levels for data quality [1], to use a framework of timestamp imperfections [2] or a framework for event log quality [3]. Better understanding of how data quality issues affect the event log quality led to the definition of so-called event log imperfection patterns [4].

To detect data quality violations this paper presents the Event log Quality Pointer (*EloqQP*) tool. The tool allows to detect event log imperfection patterns and to classify the data violations according to data quality levels as specified in the process mining manifesto [5]. Beside this, a comparison between two event logs with respect to data quality violations is supported. Thus, *ElogQP* detects missing start or end activities and activities with wrong order. Fig. 1 shows how *ElogQP* works when two event logs are used as input. The event log on the left-hand side is (more) complete, while on the right-hand side one timestamp and one activity are missing. When parsing both event logs, *ElogQP* returns data types that have been identified as data quality violations with a descriptive comment to understand the violation (see "*Output of ElogQP*").

The paper is structured as follows. Section 2 gives an overview of *ElogQP*. It describes the components and the functionality of the tool. Section 3 concludes the paper.

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

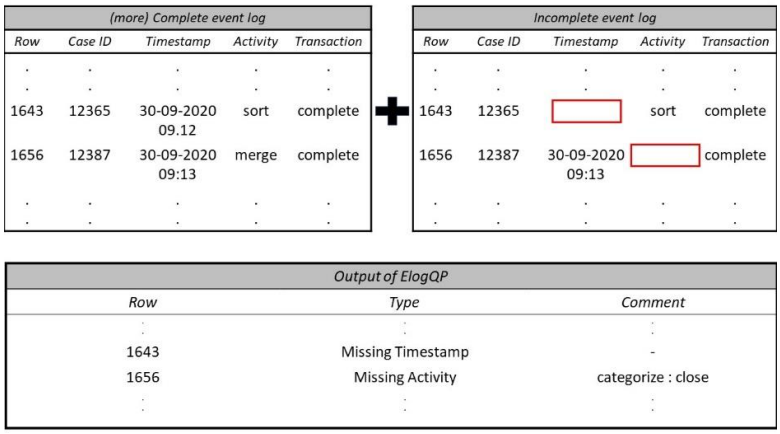


Fig. 1. *ElogQP* detects missing timestamp and missing activity.

2 Detection of Data Quality Violations

This section summarizes event log imperfection patterns and data quality levels of an event log. Section 2.2. presents how *ElogQP* refers to both.

2.1 Event Log Imperfection Patterns and Data Quality Levels

Eleven event log imperfection patterns for process mining have been defined, which are form-based event capture, inadvertent time travel, scattered event, elusive case, scattered case, collateral events, polluted label, distorted label, synonymous labels and homonymous labels. These patterns relate to data quality issues in timestamps, case IDs, activities and activity labels like missing or incorrect activities, missing case IDs and discrepancies in the activity names.

According to the process mining manifesto [5] five quality levels exist for event logs. Quality level 1 means that the recorded events do not exist in reality and thus the event log has artificial events. Often these are manually created event logs. Quality level 2 refers to event logs that are recorded without a systematic approach. This returns log data that is incorrect or incomplete. Event logs with a quality level 3 are reliable in a way that the recorded event data is likely to correspond with reality. Quality level 4 means that event logs are complete in terms of "correct". Quality level 5 fulfills the properties of quality level 4. Additionally, the recorded events have clear semantics and are well defined. *ElogQP* evaluates quality violations according to quality level 1 to 4.

2.2 Tool Overview

Fig. 3 shows the functionality of the *ElogQP* tool. The tool has been implemented in R and in essence, the tool represents a script with the following sequential steps:

- *Step 1*: The event log is imported in XES format into the *ElogQP* environment.
- *Step 2 (a)*: The user selects the event log quality attributes to be analyzed.

- *Step 2 (b)*: An additional event log or Petri net can be used as input. The comparison between the Petri net and an event log additionally allows detecting activity order incompliance. With the additional event log missing attributes can be detected.
- *Step 3*: The event log is analyzed according to the selected attributes.
- *Output*: If any data quality issue is found, *ElogQP* sets a pointer, indicates the data quality level and returns a descriptive comment as shown exemplary in Fig. 1.

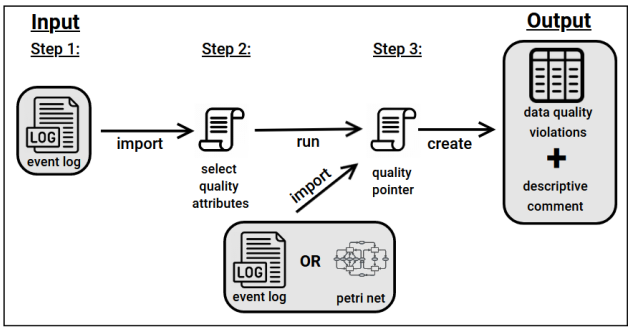


Fig. 3. How *ElogQP* works

Fig. 4 shows the output of *ElogQP* with a quality level of 2 and the detected data quality violations. If no data quality violations are found, a quality level of 4 is returned.

	Row	Typ	Comment
		Event log with quality 2	
1430	6	Missing Activity	check vacation area : Determine budget
1431	7	Missing Activity	Determine budget : determine weather
1432	8	Missing Activity	determine weather : Determine holiday type
1433	9	Missing Activity	Determine holiday type : Ask employer for vacation
1434	10	Missing Activity	Ask employer for vacation : Determine desired destination
1435	11	Missing Activity	Determine desired destination : Ask employer for vacation
1436	13	Missing Activity	Determine duration : Send offers
1437	14	Missing Activity	Send offers : Check form
1	15	Wrong Name	Check form
1438	15	Missing Activity	Check form : Wait for corrected form
2	16	Wrong Name	Wait for corrected form
1439	16	Missing Activity	Wait for corrected form : Check form
1440	17	Missing Activity	Check form : Wait for corrected form
1441	18	Missing Activity	Wait for corrected form : Logout
1443	19	Missing End	Logout
1442	20	Missing Start	register
1444	23	Missing Activity	Determine budget : check vacation area
1445	24	Missing Activity	check vacation area : Check criteria

Fig. 4. Interface of *ElogQP*

3 Conclusion and Future Work

This paper presented *ElogQP*, a tool to inspect event logs in order to find data quality violations in terms of event log imperfection patterns. In this way, *ElogQP* is a tool for cleaning event logs thus improving the process discovery result. In future work we plan to completely implement all event log imperfection patterns. So far, *ElogQP* does not detect unanchored events, elusive case and scattered case. Additionally, we will integrate data quality recommendations that have be suggested for process activity labels [6] into *EloqQP*.

References

1. Leemans, M., van der Aalst, W.M.P.: Discovery of frequent episodes in event logs. In: SIMPDA 2014: 31-45, vol. 1293 of CEUR Workshop Proceedings
2. Fischer, D. A., Goel, K., Andrews, R., Dun, C. G. J. van, Wynn, M.T., Röglinger, M.: Enhancing Event Log Quality: Detecting and Quantifying Timestamp Imperfections. BPM 2020, vol. 12168 of LNCS, Springer, pp. 309-326.
3. Kherbouche, O. M., Laga, N., Masse, P.-A. (2016): Towards a better assessment of event logs quality. SSCI 2016, IEEE, pp. 1-8.
4. Suriadi, S., Andrews, R., Hofstede, A.H.M. ter, Wynn, M.T. (2017): Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems 64: 132-150: https://doi.org/10.1016/j.is.2016.07.011.
5. van der Aalst, W.M.P. et al. (2012) Process Mining Manifesto. Business Process Management Workshops (1) 2011: 169-194, https://doi.org/10.1007/978-3-642-28108-2_19.
6. Koschmider, A., Ullrich, M., Heine, A., Oberweis, A. (2015): Revising the Vocabulary of Business Process Element Labels. CAiSE 2015, vol. 9097 of LNCS, Springer, pp. 69-8.

Analysis of Prevalent BPMN Layout Choices on GitHub

Daniel Lübke[1,2] [https://orcid.org/0000-0002-1557-8804] and Daniel Wutke[3]

[1] Digital Solution Architecture, Hanover, Germany
[2] Leibniz Universität Hannover, Germany
daniel.luebke@digital-solution-architecture.com
https://www.digital-solution-architecture.com
[3] dwutke@gmail.com

Abstract. Layout of BPMN diagrams greatly influences their understandability. The primary objective of this study is to understand prevalent choices of modelers for their design of BPMN diagrams. As a research method we use repository mining to analyze BPMN diagrams we found on GitHub. We found that BPMN diagrams on GitHub are mostly laid out from left-to-right and that layout direction choices differ by the modeling tool, process model type (pool vs. no pools) and purpose (toy vs. non-toy).

Keywords: BPMN Layout · GitHub Mining · Repository Mining

1 Introduction

Layout is one of the influencing factors of understandability of BPMN diagrams [3]. While some empirical research exists on this topic, we want to explore real-world BPMN processes and analyze the use of layouts – and influencing factors of those; for example, what layout direction (left-right vs. top-bottom) is dominantly used?

While GitHub has been used in software engineering research [6], its use for BPMN-related research is only in the beginning [4, 5]. Within this paper we re-use the dataset by Heinze et al. [4] and present a preliminary analysis of layout direction choices made for the BPMN diagrams contained therein.

We present a preliminary study, which is structured as follows: First we present our research design in Sect. 2 before we explain how we mined GitHub and how we handled the obtained models in Sect. 3. Results of our statistical analysis are presented in Sect. 4 for which we give our interpretation in Sect. 5. Finally, we discuss threats to validity in Sect. 6 before we conclude and give possible future research topics.

2 Research Questions

We want to answer the following research questions related to the layout direction of BPMN diagrams found on GitHub:

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

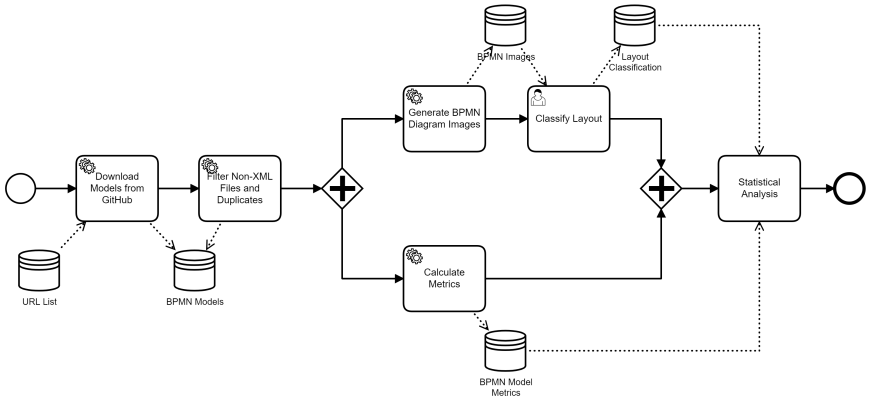


Fig. 1: The Research Process Followed

RQ1: What layout directions are used and how is their usage distributed?
Because existing research predicts better understandability for horizontal layouts [2, 3] and existing modeling guidelines mandate them [1, 9], we hypothesize that left-right diagrams are in the majority.

RQ2: What influence does the modeling tool have on layout direction?
Differences due to modeling tools have been shown for BPEL [8]. We expect that such differences also exist with BPMN.

RQ3: What influence has project ownership on layout direction?
We expect that (explicit or implicit) modeling guidelines and shared authorship within a project will lead to uniformity of layouts in any given project. Thus, we expect that the majority of diagrams within a project will have the same layout.

RQ4: Are "toy"diagrams laid out differently?
Because it has been demonstrated before that models exhibit different properties based on different purposes [7] (e.g., productive vs. example), we expect toy diagrams to be smaller and thus to have simpler layouts (horizontal & vertical only).

RQ5: Are diagrams with pools laid out differently?
Because we expect that laying out pools with more complex layouts is difficult, we expect pools to have significantly more left-right and top-bottom layouts. Because we think that pools lead to even more left-right modeling, we expect that the proportion of this layout is even larger in the diagrams with pools.

In order to answer our research questions we followed the following research process as illustrated in Fig. 1: We downloaded all files in the list of [4] (not all of which were still available online) and filtered those files according to the steps described in Sect. 3.

Later, both authors manually classified the layout of the diagrams and calculated BPMN metrics with a self-written tool called *BPMN Layout Analyzer* for various diagram metrics[4]. All classification data and metric data was stored in CSV files on which statistical analysis was performed with R. These steps are described in Sect. 4.

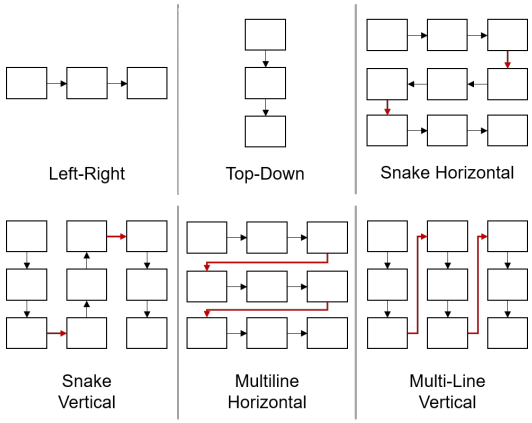


Fig. 2: Names of Layout Directions

During analysis of layout directions, we labeled BPMN diagrams as shown in Fig. 2.

3 GitHub Mining and Data Cleansing

We started by downloading the BPMN files from GitHub as they have been identified by Heinze et al. [4]. This means that we did not mine GitHub per se but downloaded all models by the list of models identified by Heinze et al. Although the original list contained 8904 unique BPMN files, only 8467 files were still available as of 2020-04-06.

For each diagram we generated a PNG file by using BPMN.io's bpmn-to-image. This failed for some files due to missing diagram interchange information or other file format compliance issues. This left us with 5299 unique processes.

In addition BPMN DI layout information or the XML itself were corrupt, which was ignored by BPMN.io so that after merging in the results from our BPMN Layout analyzer tool, only 4638 diagrams were left. In order to exclude "junk diagrams" we removed diagrams from the data set, which did not have a) at least two activities (neither counting events nor gateways), and b) were

[4] Freely available at: `https://github.com/dluebke/bpmnlayoutanalyzer/`

Table 1: Distribution of Diagram Layouts

Layout	% total	% non-toy	% toy	% no pools	% pools
left-right	79.52	73.35	86.30	78.21	86.80
multiline-horizontal	0.55	0.67	0.42	0.41	1.32
multiline-vertical	0.10	0.19	0.00	0.12	0.00
other	9.34	10.07	8.54	9.12	10.56
snake-horizontal	1.96	2.88	0.95	2.19	0.66
snake-vertical	0.20	0.29	0.11	0.24	0.00
top-down	8.33	12.56	3.69	9.71	0.66

connected enough. Too low connectedness is found in diagrams that are just used for placing all BPMN elements without any sequence flows, which was probably done to make illustrations.

Therefore, we used the following threshold for the number of subgraphs sg:

$$p = \begin{cases} 2 \times |pools^{expanded}| & : pools^{expanded} > 0 \\ 2 & : pools^{expanded} = 0 \end{cases} \tag{1}$$

$$s^e = |subprocesses^{expanded}| \tag{2}$$

$$e^e = 2 \times |eventsubprocess^{expanded}| \tag{3}$$

$$e^c = |eventsubprocess^{collapsed}| \tag{4}$$

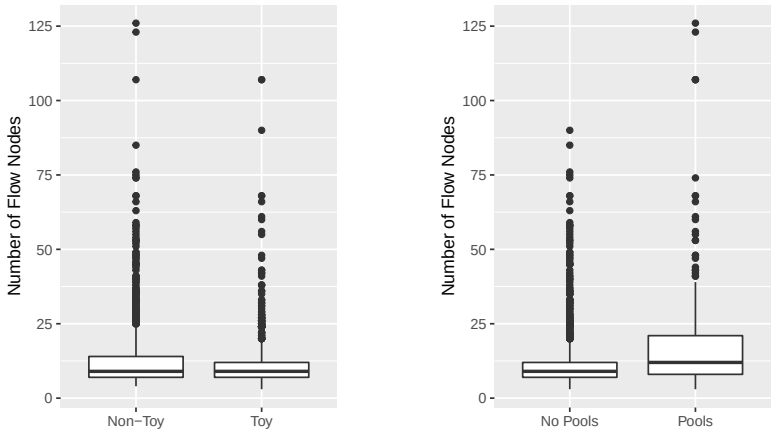
$$sg \leq p + s^e + e^e + e^c \tag{5}$$

First we define how many subgraphs our process-flow is allowed to have (equation 1): We want to exclude diagrams in which the main process falls apart into more than 2 subgraphs. For diagrams with pools we allow 2 subgraphs per expanded pool. Next, we allow an additional subgraph for an expanded subprocess (equation 2) because a new process must be contained in it. Event subprocesses are different because they are not connected to the main process flow. As such, an additional subgraph must be granted for each event subprocess, if the event subprocess is collapsed (equation 4). If the event subprocess is expanded two additional subgraphs are allowed: one for the event subprocess and one for the process contained in it (equation 3).

This left us with 1992 diagrams for analysis. The analysis of the influence of project ownership on layout directions includes duplicates and is based on a total of 7396 processes and 2745 processes after determining metrics and relevance filtering.

4 Execution & Statistics

For getting process and layout direction counts both authors manually and independently classified the diagram layout direction. After the first round, approx. 10% differences between layout direction classifications had been found which were resolved later in a shared session to reach a unified understanding.



(a) Toy vs. No-Toy (b) Pools vs. No Pools

Fig. 3: Comparison of Flow Node Count for different Diagram Subsets

The total distribution of layout directions is shown in the first data column in Table 1: The most common layout direction was left-to-right, followed by "other" layouts, which describe chaotic or too unclean layout directions, and top-down layouts. More advanced layouts like snake or multi-line layouts are rarely used.

We further classified if a BPMN model is a "toy" model or not by searching for the key words "test", "dummy", and "example" in the complete file name including path. The distribution of layout directions with regards to toy vs. non-toy processes are shown in the middle columns of Table 1. Left-right layouts are used even more frequently in toy diagrams, while top-down layouts are used more often with non-toy diagrams. We performed a simulated Fisher's Exact Test (100,000 rounds) to test whether the distributions of layout directions is significantly different between the toy and non-toy subsets.

In the following we calculated the number of pools and flow nodes in the processes with the *BPMN Layout Analyzer*. The distribution of layout directions for BPMN diagrams with or without pools are shown in the last two columns in Table 1. There are more left-right layouts used in conjunction with pools than without and more top-down layouts are used without pools than with pools. We again used a simulated Fisher's Exact Test (100,000 rounds) in order to check whether the distributions of layout directions is significantly different between diagrams with or without pools. This test again yields a highly significant p-value ($p = 9.999\,9 \times 10^{-6}$).

In a next step we analyzed the sizes of diagrams measured in number of flow nodes for toy vs. no toy diagrams (see Fig. 3a) and for diagrams with or without pools (see Fig. 3b): The mean number of flow nodes of the toy subset is 11.24 and the mean of the non-toy subset is 13.4. A Wilcoxon hypothesis test for a

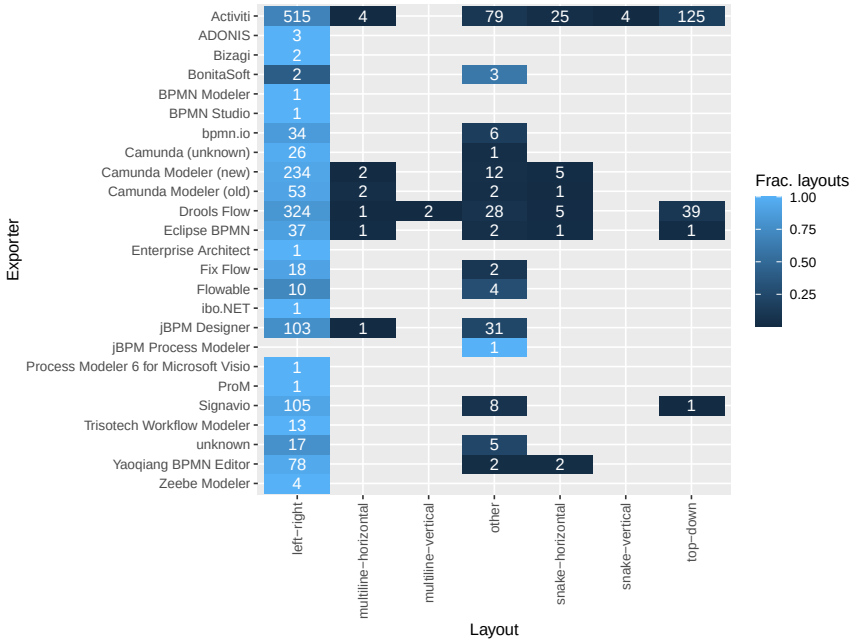


Fig. 4: Absolute and Relative Numbers of Processes by Layout and BPMN Editor

difference of means yields a p-value of 0.013 1. The mean number of flow nodes of the pools subset is 18.08 and 11.34 for diagrams without pools. A Wilcoxon hypothesis test for a difference of means yields a highly significant p-value of 1.514×10^{-19}.

The "BPMN Layout Analyzer" tool also extracts the exporter meta data (which describe the BPMN tool that wrote that file) from BPMN files. When no exporter meta data was found, some heuristics, e.g., namespace names, were used to find a potential BPMN editor. However, there are still some ambiguities, e.g., Camunda and bpmn.io have different names and we do not know for sure whether these name changed in different releases or whether the exporter info was set incorrectly by some other BPMN tool.

We broke down the number of diagrams grouped by layout direction and the BPMN editor as shown in Fig. 4: Nearly all tools have left-right or other layouts only with small exceptions. However, both Activiti and Drools also have a large number of top-down layouts. They are practically the only editors, which have been used to create top-down layouts, although the majority of diagrams created with these tools still follow a left-right layout. We performed a simulated Fisher's Exact Test (100,000 rounds) to test whether the distribution of layout directions is independent from the modeling tool used. This test yields a highly significant p-value of $p = 9.999\,9 \times 10^{-6}$.

Lastly, we analyzed the layout direction "cleanliness" for the repositories. For each repository we calculated the most dominant layout direction for all diagrams contained therein. Then we calculated the percentage of diagrams that have this layout direction compared to all diagrams within this repository. Thus, cleanliness of 100% means that all diagrams in such a repository have the same layout direction. 85.02% of all repositories had the same layout direction for all their diagrams. Furthermore, we performed a simulated Fisher's Exact Test (100,000 rounds) for the different layout direction distributions against the repositories, which yields a highly significant p-value of $p = 9.999\,9 \times 10^{-6}$.

5 Interpretation

RQ1: What layout directions are used and how is the usage distributed?
We found that the majority (79.52 %) of diagrams are laid out left-right. Although we do not know what the causal reasons are, the left-right layout is predominantly used as recommended by theory, existing guidelines, and the BPMN specification.
RQ2: What influence does the modeling tool have on layout direction?
The hypothesis test is highly significant indicating that the modeling tool has an impact on the diagram layout direction. Interestingly, the Activiti and Drools modelers are responsible for nearly all top-down layouts. However, it is unclear at this point, why these tools have been used for top-down layouts, which warrants further investigation. Many editors have preference for left-right layouts, e.g., Camunda and Signavio. Thus, investigating editor preferences and linking them to actually used layouts can possibly give more insights.
RQ3: What influence does the project ownership has on layout direction?
Layout directions differ highly significantly depending on the project ownership, i.e., the owning repository. While we could not dive deep into the data yet, the differences are highly significant: 85.02% of the repositories followed only one layout paradigm; others had diagrams with different layouts. This means that there are forces which will make diagrams in a projects more similar. Future research can investigate what those forces are (e.g., same developers, guidelines, ...).
RQ4: Are "toy"diagrams laid out differently?
Within the dataset toy diagrams have a highly significantly different layout distribution and are highly significantly smaller with regards to their flow node count. As such we conclude that "toy" diagrams are not representative for the set of "non-toy" BPMN diagrams and future research should be concerned whether to include or exclude those depending on the research questions.
RQ5: Are diagrams with pools laid out differently?
Within the dataset diagrams with pools have a highly significantly different layout distribution and are highly significantly larger with regards to their flow node count. As such we conclude that for future research into BPMN models and diagrams, pool and non-pool diagrams should be researched separately.

Because this is a exploratory study based on existing data without any control, all these correlations can be due to confounding reasons or because they are really causal. Further research is required to establish the relationship type.

6 Threats to Validity

Like in software engineering research a general threat is the usage of GitHub data that might not be representative and generalizable [6]. In fact, we have shown that further analysis must take care of diagram types. Also, due to manual nature of the layout classification other researchers might come to other results. We also experienced problems with the diagram interchange information in the BPMN files, which can skew the results to more reflect compliant editors. Lastly, the tool distribution found on GitHub does not match those found in organizations (e.g., IBM, SAP, and Oracle tools are missing; Signavio is underrepresented etc.). Some BPMN models had more than 1 diagram, which we did not evaluate. On a technical note, the exporter information in the BPMN diagrams itself are not as reliable as one would hope: Missing information and ambiguous names are possible sources of error.

7 Conclusion & Outlook

Within this paper we have shown that the majority of BPMN diagrams found on GitHub are laid out left-right. The results suggest that the type ("toy" or "non-toy") of a process model influences the size and the layout direction. We have found that further research into tool usage is warranted as nearly all top-down diagrams are laid out using only two editors. Furthermore, we have shown that most – although not as many as expected – repositories only contain diagrams with one layout.

This work opens up new research angles: 1) How can "real" models be separated from "toy" models automatically? Our heuristics of using key words in the file path is a first approximation but while manually classifying the layouts, we also encountered other diagrams (empty or default labels, unfinished diagrams, ...) that should possibly excluded. 2) The exact causal relationships between model properties (size, pools), modeling tooling and the diagram layout needs to be researched. We showed correlations but no causal relationships in this work. 3) This study should be replicated and compared to model repositories from larger organizations created by process modelers in their respective environments. 4) Formalization of layout direction and automation of its detection in order to scale to larger datasets and make the classification more objective. All in all, this explorative work has laid the foundations for answering these questions.

Bibliography

1. Angela Birchler, Elisabeth Bosshart, Mike Märki, Peter Opitz, Jürg Pauli, Beat Rigert, Yves Sandoz, Marc Schaffroth, Nicki Spöcker, Christian Tanner, Konrad Walser, and Thomas Widmer. eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung. WWW: https://www.ech.ch/dokument/fb5725cb-813f-47dc-8283-c04f9311a5b8, September 2014.
2. Kathrin Figl and Mark Strembeck. On the importance of flow direction in business process models. In *2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA)*, pages 132–136. IEEE, 2014.
3. Kathrin Figl and Mark Strembeck. Findings from an experiment on flow direction of business process models. In *EMISA 2015*, 2015.
4. Thomas Heinze, Viktor Stefanko, and Wolfram Amme. Bpmn in the wild: Bpmn on github. com. In *Proceedings of the 12th ZEUS Workshop on Services and their Composition*, pages 26–29. CEUR-ws. org, 2020.
5. Thomas S. Heinze, Viktor Stefanko, and Wolfram Amme. Mining bpmn processes on github for tool validation and development. In Selmin Nurcan, Iris Reinhartz-Berger, Pnina Soffer, and Jelena Zdravkovic, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 193–208, Cham, 2020. Springer International Publishing.
6. Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 92–101, New York, NY, USA, 2014. Association for Computing Machinery.
7. Daniel Lübke, Ana Ivanchikj, and Cesare Pautasso. A template for categorizing empirical business process metrics. In *Business Process Management Forum - BPM Forum 2017*, 2017.
8. Daniel Lübke, Tobias Unger, and Daniel Wutke. Analysis of data-flow complexity and architectural implications. In Daniel Lübke and Cesare Pautasso, editors, *Empirical Studies on the Development of Executable Business Processes*, pages 59–80. Springer, 2019 (to be published).
9. Bruce Silver and Bruce Richard. *BPMN Method and Style*, volume 2. Cody-Cassidy Press Aptos, 2009.

A Deep Q-learning Scaling Policy for Elastic Application Deployment

Fabiana Rossi

Department of Civil Engineering and Computer Science Engineering
University of Rome Tor Vergata, Italy
f.rossi@ing.uniroma2.it

Abstract The ability of cloud computing to provide resources on demand encourages the development of elastic applications. Differently from the popular threshold-based solutions used to drive elasticity, we aim to design a flexible approach that can customize the adaptation policy without the need of manually tuning various configuration knobs. In this paper, we propose two Reinforcement Learning (RL) solutions (i.e., Q-learning and Deep Q-learning) for controlling the application elasticity. Although Q-learning represents the most popular approach, it may suffer from a possible long learning phase. To improve scalability and identify better adaptation policies, we propose Deep Q-learning, a model-free RL solution that uses a deep neural network to approximate the system dynamics. Using simulations, we show the benefits and flexibility of Deep Q-learning with respect to Q-learning in scaling applications.

Keywords: Deep Q-learning · Elasticity · Reinforcement Learning · Self-adaptive systems.

1 Introduction

The dynamism of working conditions calls for an elastic application deployment, which can be adapted in face of changing working conditions (e.g., incoming workload) so to meet stringent Quality of Service (QoS) requirements. Cloud providers, e.g., Amazon, Google, that support multi-component applications usually use static thresholds on system-oriented metrics to carry out the adaptation of each application component. As shown in [11,12], the manual tuning of such scaling thresholds is challenging, especially when we need to specify critical values on system-oriented metrics and the application exposes its requirements in terms of user-oriented metrics (e.g., response time, throughput, cost). Differently from the popular static threshold-based approaches, we aim to design a flexible policy that can adapt the application deployment, according on user-defined goals, without the need of manually tuning various configuration knobs.

In this paper, we use Reinforcement Learning (RL) to adapt the application deployment. RL allows to express *what* the user aims to obtain, instead of *how* it should be obtained (as required by threshold-based policies). Most of the existing works consider model-free RL algorithms, e.g., Q-learning, to drive the

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

application deployment (e.g., [2,5]). However, one of the main issue with Q-learning is the possibly long learning phase, which is especially experienced when the number of system states increases. An approach to boost the learning process is to provide the agent with system knowledge. By exploiting it, the agent can more easily find a suitable trade-off between system- and user-oriented metrics (i.e., resource utilization and target response time). Therefore, together with Q-learning, we propose Deep Q-learning [8]. It uses deep neural networks to approximate the system dynamics and to drive the application elasticity. Using simulation, we demonstrate the advantages of the Deep Q-learning solution, which learns a suitable scaling policy while meeting QoS requirements expressed in term of target application response time (i.e., $R_{\max}$).

2 Related Work

The existing elasticity solutions rely on a wide set of methodologies, that we classify in the following categories: mathematical programming, control theory, queuing theory, threshold-based, and machine-learning solutions. The mathematical programming approaches exploit methods from operational research in order to solve the application deployment problem (e.g., [13,21]). The main drawback of these solutions is scalability; since the deployment problem is NP-hard, other efficient solutions are needed. As surveyed in [15], few solutions use control theory to change the replication degree of application containers (e.g., [3]). The critical point of the control-theoretic approaches is the requirement of a good system model, which can sometimes be difficult to be formulated. The key idea of queuing theory is to model the application as a queuing system with inter-arrival and service times having general statistical distributions (e.g., [4,10]). Nevertheless, queuing theory often requires to approximate the system behavior, discouraging its adoption in a real environment. Most of the existing solutions exploit best-effort threshold-based policies based on the definition of static thresholds for adapting the application deployment at run-time (e.g., [7,16]). Although a threshold-based scaling policy can be easily implemented, it is a best-effort approach that moves complexity from determining the reconfiguration strategy to the selection of critical values that act as thresholds. In the last few years, machine learning has become a widespread approach to solve at run-time the application deployment problem. RL is a machine learning technique by which an agent can learn how to make good (scaling) decisions through a sequence of interactions with the environment [17]. After executing an adaptation action in the monitored system state, the RL agent experiences a cost that contributes to learning how good the performed action was. The obtained cost leads an update of a lookup table that stores the estimation of the long-term cost for each state-action pair (i.e., the Q-function). Many solutions considered the classic model-free RL algorithms (e.g., [9,18]), which however suffer from slow convergence rate. To tackle this issue, different model-based RL approaches have been proposed, e.g., [14,19]. They use a model of the system to drive the action exploration and speed-up the learning phase. Although model-based RL approaches

can overcome the slow convergence rate of model-free solutions, they can suffer from poor scalability in systems with a large state space. In this solution, the lookup table has to store a separate value for each state-action pair. An approach to overcome this issue consists in approximating the system behavior, so that the agent can explore a reduced number of system configurations. Integrating deep neural networks into Q-learning, Deep Q-learning has been widely applied to approximate the system dynamics in a variety of domains, e.g., traffic offloading [1]. However, to the best of our knowledge, it is so far poorly applied in the context of application elasticity. Differently from all the above contributions, in this paper we propose an application scaling policy based on Deep Q-learning; then, we compare it against Q-learning.

3 RL-based Scaling Policy

RL is a special method belonging to the branch of machine learning. It refers to a collection of trial-and-error methods by which an agent must prefer actions that it found to be effective in the past (*exploitation*). However, to discover such actions, it has to explore new actions (*exploration*).

At each discrete time step i, according to the monitored metrics, the RL agent determines the application state and updates the expected long-term cost (i.e., Q-function). We define the application state as $s = (k, u)$, where k is the number of application instances, and u is the monitored CPU utilization. We denote by $\mathcal{S}$ the set of all the application states. We assume that $k \in \{1, 2, ..., K_{\max}\}$; being the CPU utilization (u) a real number, we discretize it by defining that $u \in \{0, \bar{u}, ..., L\bar{u}\}$, where $\bar{u}$ is a suitable quanta and $L \in \mathbb{N}$ such that $L \cdot \bar{u} = 1$. For each state $s \in \mathcal{S}$, we define the set of possible adaptation actions as $\mathcal{A}(s) \subseteq \{-1, 0, 1\}$, where $+1$ is the *scale-out* action, -1 the *scale-in* action, and 0 is the *do nothing* decision. We observe that not all the actions are available in any application state, due to the lower and upper bounds on the number of application instances (i.e., 1 and $K_{\max}$, respectively). According to an action selection policy, the RL agent identifies the scaling action a to be performed in state s. The execution of a in s leads to the transition in a new application state (i.e., s') and to the payment of an immediate cost. We define the immediate cost $c(s, a, s')$ as the weighted sum of different normalized terms, such as the *performance penalty*, c_{perf}, and *resource cost*, c_{res}. Formally, we have:

$$c(s, a, s') = w_{\text{perf}} \cdot c_{\text{perf}} + w_{\text{res}} \cdot c_{\text{res}} \tag{1}$$

where w_{perf} and w_{res}, $w_{\text{perf}} + w_{\text{res}} = 1$, are non negative weights that allow us to express the relative importance of each cost term. The *performance penalty* is paid whenever the average application response time exceeds the target value $R_{\max}$. The *resource cost* is proportional to the number of application instances. We can observe that the formulation of the immediate cost function $c(s, a, s')$ is general enough and can be easily customized with other QoS requirements.

The received immediate cost contributes to update the Q-function. The Q-function consists in $Q(s, a)$ terms, which represent the expected long-term cost

that follows the execution of action a in state s. The existing RL policies differ in how they update the Q-function and select the adaptation action to be performed (i.e., action selection policy) [17]. In [14], for example, we propose a model-based RL approach that enriches the RL agent with a model of the system to drive the exploration actions and speed up the learning phase. Since determining the system model can be a not trivial task, in this paper we consider two model-free RL solutions to adapt the application deployment. First, we consider the simple model-free Q-learning (QL) algorithm that uses a table (i.e., Q-table) to store the Q-value for each state-action pair. The Q-table allows to store the real experience without approximation. However, this approach may suffer from slow convergence rate when the number of state-action pairs increases. Then, to tackle this issue, we present a Deep Q-learning (DQL) approach that combines Q-learning with deep neural networks. The neural network allows to approximate the Q-function using a non-linear function; in such a way, the agent can directly compute $Q(s, a)$ using s and a, instead of performing a Q-table lookup. By using a Q-function approximation, the RL agent does not need to explore all the state-action pairs before learning a good adaptation policy.

Q-learning. At time i, the QL agent selects the action a to perform in state s using an ϵ-greedy policy on $Q(s, a)$; the application transits in s' and experiences an immediate cost c. The ϵ-greedy policy selects the best known action for a particular state (i.e., $a = argmin_{a \in A(s)} Q(s, a)$) with probability $1 - \epsilon$, whereas it favors the exploration of sub-optimal actions with low probability, ϵ. At the end of time slot i, $Q(s, a)$ is updated using a simple weighted average:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[c + \gamma \min_{a' \in \mathcal{A}(s')} Q(s', a') \right] \tag{2}$$

where $\alpha, \gamma \in [0, 1]$ are the *learning rate* and the *discount factor*, respectively.

Deep Q-learning. DQL uses a multi-layered neural network, called Q-network, to approximate the Q-function. For each time slot i, the DQL agent observes the application state and selects an adaptation action using an ϵ-greedy policy and the estimates of Q-values, as Q-learning does. Note that DQL is model-free: it solves the RL task directly using samples, without explicitly modeling the system dynamics. In a given state s, the Q-network outputs a vector of action values $Q(s, \cdot, \phi)$, where ϕ are the network parameters. By approximating the Q-function, the RL agent can explore a reduced number of system configurations before learning a good adaptation policy. At the end of each time slot i, the Q-network is updated by performing a gradient-descent step on $(y_i - Q(s, a, \phi_i))^2$ with respect to the network parameters ϕ_i. y_i is the estimated long-term cost, defined as $y_i = c + \gamma \cdot min_{a'} Q(s', a', \phi_i)$, where γ is the discount factor. When only the current experience is considered, i.e., (s, a, c, s'), this approach is too slow for practical real scenarios. Moreover, it is unstable due to correlations existing in the sequence of observations. To overcome these issues, we consider a revised DQL algorithm that uses a replay buffer and a separate target network to compute y_i [8]. To perform experience replay, the agent store its experience in a buffer with finite capacity. A mini-batch of experience is drawn uniformly at random from the replay buffer to remove correlations in the observation sequence

and to smooth over changes in the data distribution. In the classic DQL, the same Q-network is used both to select and to evaluate an action. This can lead to select overestimated values, resulting in overoptimistic estimates. To prevent this, two networks are used (i.e., on-line and target network) and two value functions are learned. The on-line network is used to determine the greedy policy and the target network to determine its value. The target network parameters are updated to the on-line network values only every τ steps and are held fixed between individual updates.

4 Results

We evaluate the proposed deployment adaptation solutions using simulations. Without lack of generality, at each discrete time step i, we model the application as an $M/M/k_i$ queue, where k_i is the number of application replicas. We set the service rate μ to 120 requests/s. As shown in Fig. 1, the application receives a varying number of requests. It follows the workload of a real distributed application [6]. The application expresses the QoS in terms of target response time $R_{\max} = 15$ ms. The RL algorithms use the following parameters: $\alpha = 0.1$ and discount factor $\gamma = 0.99$. We discretize the application state with $K_{\max} = 10$ and $\bar{u} = 0.1$. To update the application deployment, QL and DQL use an ϵ-greedy action selection policy, with $\epsilon = 0.1$. DQL uses a replay memory with capacity of 50 observations and a batch size of 30; the target Q-network update frequency is $\tau = 5$ time units. We use Deeplearning4j[1] library to implement the neural networks. Correctly configuring the Q-network is an empirical task, which requires some effort and several preliminary evaluations. In particular, we use ReLu as the neuron activation function; due to its non-linear behavior, it is one of the most commonly used function. To initialize the Q-network weights, we use the Xavier method [20]. To avoid weights to diminish or explode during network propagation, this method scales the weight distribution on a layer-by-layer basis. To this end, it uses a normal distribution with centered mean and standard deviation scaled to the number of layer's input and output neurons.

[1] https://deeplearning4j.org/

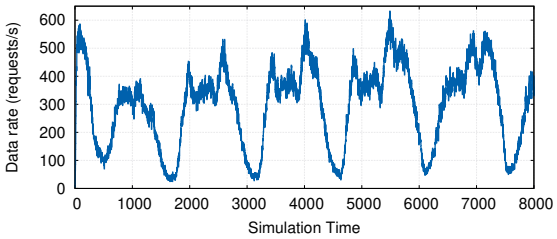
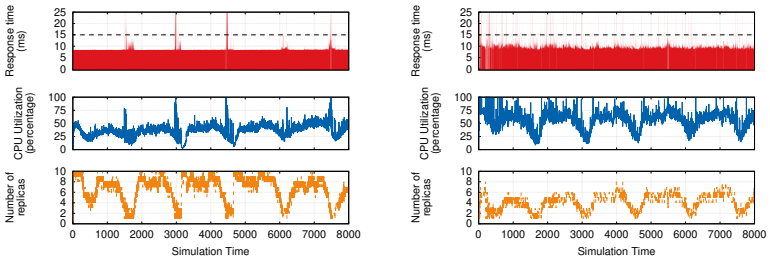


Figure 1: Workload used for the reference application.

Table 1: Application performance under scaling policies.

Elasticity Policy	*Configuration*	$R_{\max}$ *violations* (%)	*Average CPU utilization* (%)	*Average number of replicas*	*Median of Response time (ms)*
QL	$w_{\text{perf}} = 1, w_{\text{res}} = 0$	3.15	56.20	4.17	9.51
	$w_{\text{perf}} = 0.5, w_{\text{res}} = 0.5$	13.26	51.27	5.13	8.81
	$w_{\text{perf}} = 0, w_{\text{res}} = 1$	42.27	65.56	3.90	11.88
DQL	$w_{\text{perf}} = 1, w_{\text{res}} = 0$	1.05	36.85	6.63	8.38
	$w_{\text{perf}} = 0.5, w_{\text{res}} = 0.5$	39.23	66.64	3.58	11.11
	$w_{\text{perf}} = 0, w_{\text{res}} = 1$	88.51	89.18	1.58	$+\infty$
DQL with pre-trained network	$w_{\text{perf}} = 1, w_{\text{res}} = 0$	1.39	35.54	7.41	8.35
	$w_{\text{perf}} = 0.5, w_{\text{res}} = 0.5$	31.91	58.53	4.30	9.06
	$w_{\text{perf}} = 0, w_{\text{res}} = 1$	83.48	86.12	1.71	$+\infty$

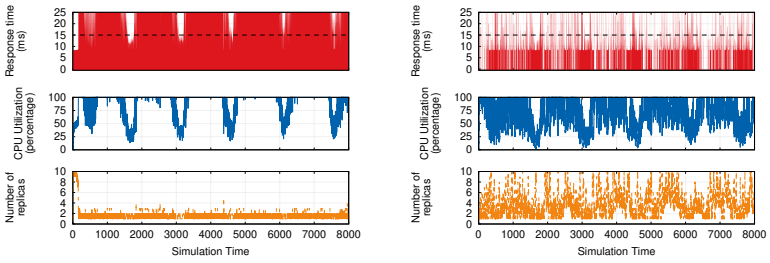


(a) Deep Q-learning (b) Q-learning

Figure 2: Application performance using the weights $w_{\text{res}} = 0$ and $w_{\text{perf}} = 1$.

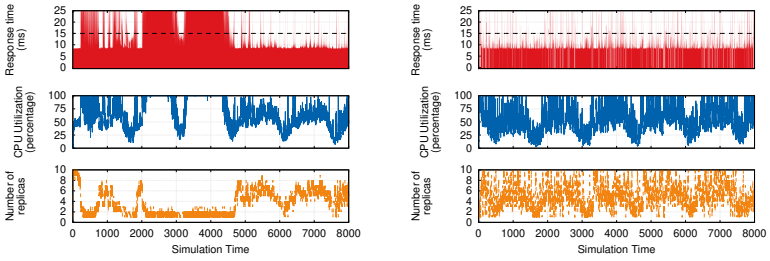
The Q-network architecture is fully-connected with 4 layers having $\{2, 15, 15, 3\}$ neurons (i.e., there are 2 hidden layers).

Table 1 summarizes the experimental results, including also the application performance obtained with a pre-trained DQL. We can see that the application has a different performance when different weights for the cost function are used (Eq. 1). We first consider the set of weights $w_{\text{perf}} = 1$ and $w_{\text{res}} = 0$: in this case, optimizing the application response time is more important than saving resources. As shown in Fig. 2b, the QL solution often changes the application deployment performing scaling operations. Moreover, the application response time exceeds $R_{\max}$ for 3.15% of the time. Conversely, taking advantage of the approximated system knowledge, the DQL solution learns a better elasticity policy that successfully controls the application deployment (Fig. 2a). It registers 1.05% of $R_{\max}$ violations and a median of the application response time lower than the target application response time (i.e., 8.38 ms). We now consider the case when saving resources is more important than meeting the $R_{\max}$ bound, i.e., $w_{\text{res}} = 1$ and $w_{\text{perf}} = 0$. Intuitively, the RL agent should learn how to improve resource utilization at the expense of a high application response time (i.e., that exceeds $R_{\max}$). Table 1 and Fig. 3 show that, in general, DQL performs better than QL in terms of resource usage. DQL registers 89.18% of resource utilization running with 1.58 application replicas. This is very close to the lowest amount of



(a) Deep Q-learning (b) Q-learning

Figure 3: Application performance using the weights $w_{\text{res}} = 1$ and $w_{\text{perf}} = 0$.



(a) Deep Q-learning (b) Q-learning

Figure 4: Application performance using the weights $w_{\text{res}} = 0.5$ and $w_{\text{perf}} = 0.5$.

resources assignable to the application. Run-time adaptations are also avoided. As a consequence, the application is overloaded and the resulting median response time is unbounded. Conversely, the QL solution struggles to find a stable configuration. It identifies an adaptation policy that runs the application using, on average, 3.90 instances. On average, its resource usage is lower than in DQL (65.56% and 89.18%, respectively), as also the percentage of the target application response time $R_{\max}$ violations. Besides the weight configurations at the opposite ends, we can obtain a wide set of adaptation strategies that differ by the relative importance of the two deployment goals. In Table 1 we propose a simple case, where we set $w_{\text{perf}} = w_{\text{res}} = 0.50$. To visualize the update of the application deployment by the two RL policies, we report in Fig. 4 the application behavior during the whole experiment, when we want to optimize the performance avoiding resource wastage ($w_{\text{perf}} = w_{\text{res}} = 0.5$). Intuitively, the RL agent has to find a trade-off between the resource usage and the number of $R_{\max}$ violations. The neural network allows to approximate the Q-function using a non-linear function; in such a way, DQL can explore a reduced number of system configurations before learning a good adaptation policy. On average, it runs the application with 3.58 replicas, registering a 66.64% of CPU utilization. The median appli-

cation response time is 11.11 ms, with about 39% of $R_{\max}$ violations. Although we could pre-train the Q-network to further improve the DQL learned policy (mitigating also the initial exploration phase), we observe that the obtained results are already remarkable, considering that DQL is model-free. Conversely, when QL updates the application deployment, it continuously performs scaling actions, meaning that QL is still exploring the best actions to perform. This behavior is also reflected on the number of application instances, whose average value is greater than those used in the $w_{\text{perf}} = 1$ configuration. Being model-free and storing experience without approximation, QL cannot quickly learn a suitable adaptation strategy for intermediate cost weight configurations during the experiment.

Discussion. In this paper, we evaluated QL and DQL to adapt the application deployment at run-time. First, we showed the flexibility provided by a RL-based solution for updating the application deployment. By correctly defining the relative importance of the deployment objectives through the cost function weights in Eq. 1, the RL agent can accordingly learn a suitable application deployment strategy. Very different application behavior can be obtained when we aim to optimize the application response time, resource saving, or a combination thereof. Second, we showed that a DQL approach takes advantage of the approximate system knowledge and outperforms QL, especially when we pre-train the Q-network. We observe that, although we do not need to define the system model as in a model-based approach, DQL introduces the effort of defining a suitable Q-network architecture. However, this is an empirical process that may require a large number of preliminary experiments and trial-and-error repetitions.

5 Conclusion

Most policies for scaling applications resort on threshold-based heuristics that require to express how specific goals should be achieved. In this paper, aiming to design more flexible solution, we have proposed Q-learning and Deep Q-learning policies for controlling the application elasticity. Relying on a simulation-based evaluation, we have shown the benefits of the proposed RL-based approaches. Deep Q-learning exploits deep neural networks to approximate the system dynamics, estimated through system interactions. The deep neural network speeds up the learning phase, improving the application performance; however, modeling the neural network architecture can be challenging.

As future work, we plan to further investigate RL approaches for elasticity. We will investigate more sophisticated techniques for improving the convergence speed of the learning process (e.g., by leveraging Bayesian Decision Trees, Function Approximation). Moreover, we plan to extend our model by explicitly considering multiple system-oriented metrics within the adaptation policies.

References

1. Alam, M.G.R., Hassan, M.M., Uddin, M.Z., Almogren, A., Fortino, G.: Autonomic computation offloading in mobile edge for IoT applications. Future Gener. Comput.

Syst. **90**, 149 – 157 (2019)
2. Arabnejad, H., Pahl, C., Jamshidi, P., Estrada, G.: A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In: Proc. of IEEE/ACM CCGrid '17. pp. 64–73 (2017)
3. Baresi, L., Guinea, S., Leva, A., Quattrocchi, G.: A discrete-time feedback controller for containerized cloud applications. In: Proc. of ACM SIGSOFT FSE '16. pp. 217–228. ACM (2016)
4. Gias, A.U., Casale, G., Woodside, M.: Atom: Model-driven autoscaling for microservices. In: Proc. of IEEE ICDCS '19. pp. 1994–2004 (2019)
5. Horovitz, S., Arian, Y.: Efficient cloud auto-scaling with SLA objective using Q-learning. In: Proc. of IEEE FiCloud '18. pp. 85–92 (2018)
6. Jerzak, Z., Ziekow, H.: The debs 2015 grand challenge. In: Proc. ACM DEBS'15. pp. 266–268 (2015)
7. Kwan, A., Wong, J., Jacobsen, H., Muthusamy, V.: Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres. In: Proc. of IEEE ICDCS '19. pp. 80–90 (2019)
8. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., et al.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
9. Nouri, S.M.R., Li, H., Venugopal, S., Guo, W., He, M., Tian, W.: Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. Future Gener. Comput. Syst. **94**, 765–780 (2019)
10. Rossi, F., Cardellini, V., Lo Presti, F.: Hierarchical scaling of microservices in Kubernetes. In: Proc. of IEEE ACSOS '20. pp. 28–37 (2020)
11. Rossi, F., Cardellini, V., Lo Presti, F.: Self-adaptive threshold-based policy for microservices elasticity. In: In Proc. of IEEE MASCOTS '20. pp. 1–8 (2020)
12. Rossi, F.: Auto-scaling policies to adapt the application deployment in Kubernetes. In: CEUR Workshop Proc. 2020. vol. 2575, pp. 30–38 (2020)
13. Rossi, F., Cardellini, V., Lo Presti, F., Nardelli, M.: Geo-distributed efficient deployment of containers with Kubernetes. Comput. Commun **159**, 161 – 174 (2020)
14. Rossi, F., Nardelli, M., Cardellini, V.: Horizontal and vertical scaling of container-based applications using Reinforcement Learning. In: Proc. of IEEE CLOUD '19. pp. 329–338 (2019)
15. Shevtsov, S., Weyns, D., Maggio, M.: Self-adaptation of software using automatically generated control-theoretical solutions. Engineering Adaptive Software Systems pp. 35–55 (2019)
16. Srirama, S.N., Adhikari, M., Paul, S.: Application deployment using containers with auto-scaling for microservices in cloud environment. J. Netw. Comput. Appl. **160** (2020)
17. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2 edn. (2018)
18. Tang, Z., Zhou, X., Zhang, F., Jia, W., Zhao, W.: Migration modeling and learning algorithms for containers in fog computing. IEEE Trans. Serv. Comput. **12**(5), 712–725 (2019)
19. Tesauro, G., Jong, N.K., Das, R., Bennani, M.N.: A hybrid Reinforcement Learning approach to autonomic resource allocation. In: Proc. of IEEE ICAC '06. pp. 65–73 (2006)
20. Xavier, G., Yoshua, B.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS'10. vol. 9, pp. 249–256 (2010)
21. Zhao, D., Mohamed, M., Ludwig, H.: Locality-aware scheduling for containers in cloud computing. IEEE Trans. Cloud Comput. **8**(2), 635–646 (2018)

Profiling Lightweight Container Platforms: MicroK8s and K3s in Comparison to Kubernetes

Sebastian Böhm and Guido Wirtz

Distributed Systems Group, University of Bamberg, Bamberg, Germany
{sebastian.boehm,guido.wirtz}@uni-bamberg.de

Abstract. Kubernetes (K8s) is nowadays the first choice for managing containerized deployments that rely on high–availability, scalability, and fault tolerance. To enable the usage of container orchestration in resource–constrained environments, lightweight distributions emerged. The platforms MicroK8s (mK8s) and K3s, which are analyzed in this paper, claim to provide an easy deployment of K8s in a simplified form and way. In terms of resource utilization and deployment time of a K8s cluster, the lightweight platforms promise savings compared to K8s. We analyzed lightweight K8s distributions in a quantitative way by performing an experiment that monitors the utilization and time consumption compared to a native K8s cluster lifecycle. This involves starting, stopping, and adding nodes as well as creating, running, and deleting deployments. We show that not all platforms exhibit a quantitative advantage over K8s. K3s caused a similar resource consumption but had some performance advantages for starting new nodes and adding nodes to the cluster. The platform MicroK8s has shown a higher resource utilization and time consumption for all steps in our modeled lifecycle simulation.

Keywords: Lightweight Kubernetes · Container orchestration · Container lifecycle · Performance model.

1 Introduction

Kubernetes (K8s), nowadays the state–of–the–art container orchestrator, enables an efficient and comfortable way to run large and complex sets of interacting containerized applications. The container platform offers a comprehensive set of features to build highly available, scalable, and fault–tolerant clusters.[1] Driven by the emergence of containerization, a lightweight way of virtualization, K8s pervaded many different application areas like Fog, Edge, and IoT computing [5–7]. Over the years, K8s was already in focus of research regarding the resource utilization during running workloads [1, 2, 4, 8, 9]. Eiermann et al. [1] have shown that K8s causes a higher utilization in idle and load conditions compared to alternative platforms like Docker Swarm. This may limit the applicability in fields like Fog, Edge, and IoT computing that are characterized by resource–constrained devices but require features like high–availability, scalability, and fault–tolerance

[1] https://kubernetes.io/

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

to work in critical areas like surveillance and smart cities. Hence, K8s distributions claiming to be lightweight emerged to leverage the former mentioned application fields with K8s–managed deployments. The platforms MicroK8s (mK8s)[2], K3s[3], KubeEdge (KE)[4], and minikube (MK)[5] provide K8s–compatible distributions by modifying and reorganizing essential components. They aim to simplify configuring, running, and maintaining clusters to enable deployments with low–end devices. So far, quantitative performance benchmarks refer mainly to idle and load conditions [1, 2]. Other studies also consider the resource consumption when creating, starting, and stopping container instances [8, 9]. Nevertheless, there is no detailed evaluation regarding the resource and time consumption of steps like starting, adding, draining, or stopping nodes. Therefore, this paper aims to propose an experimental approach to evaluate the overall lifecycle of K8s. For this, we conducted an experiment with selected platforms to answer the following research question: **What is the resource and time consumption of the lightweight distributions mK8s and K3s in comparison to native K8s during typical events in a cluster lifecycle?**

We perform a simulation in a reproducible manner to derive detailed insights regarding the resource consumption and time consumption of all platforms. We decided to select mK8s and K3s as platforms because the controlling of cluster operations, like starting and stopping nodes, is working similar to K8s. KE is not considered in our research because it requires an additional K8s in the cloud which mitigates a fair comparison to other platforms.

The rest of the paper is structured as follows: First, we discuss existing approaches to evaluate K8s distributions shortly (Section 2). Then, we cover the concepts of the considered platforms (Section 3), which help to comprehend the design and the results of the performance comparison in Section 4. Finally, we provide a critical review of the proposed experiment (Section 5) and outline plans for further experimental studies (Section 6).

2 Related Work

Benchmarking container platforms is not a novel part of research. Eiermann et al. [1] compared the CPU and memory utilization of a cluster consisting of five low–end devices running idle, running Docker Swarm, and finally K8s. Based on an HTTP load testing scenario, K8s has shown a 9–40 times higher utilization on average in comparison to Docker Swarm. Fathoni et al. [2] followed this approach and evaluated the lightweight platforms KE and K3s. They captured the CPU and memory utilization of a two–node cluster in idle and load conditions. They did not obtain a significant difference. K3s was compared to an additional K8s–compatible platform, called FLEDGE, by Goethals et al. [4]. They measured the needed amount of memory and the disk utilization of FLEDGE, K8s, and

[2] https://microk8s.io/
[3] https://k3s.io/
[4] https://kubeedge.io/en/
[5] https://minikube.sigs.k8s.io/docs/start/

K3s with different processor architectures and container runtimes. FLEDGE used around 50 % less resources than K8s and 10 % less than K3s on a x64 architecture. Medel et al. [8, 9] investigated different operational states of pods and containers. They measured the time to create, execute, restart, and stop a varying number of containers managed by a certain amount of pods. In addition, they obtained metrics for CPU, memory, disk, and network utilization.

However, there is no comprehensive analysis how the different platforms perform during a complete cluster lifecycle. Former work focuses mainly on idle and load conditions as well as applying deployments to an already existing cluster. The resource consumption for operations like starting, stopping, adding and removing nodes from the cluster is not considered. Hence, we provide a performance comparison model that covers all steps to track creating, starting, running, stopping, and deleting of a K8s cluster.

3 Lightweight Kubernetes

Kubernetes. The container platform K8s represents a cluster based on a set of worker nodes. The worker nodes run so–called pods that contain a set of workloads (e.g., applications or batch jobs) to be executed. The set of worker nodes is managed by the *control plane*, which consists of several components that can be distributed over different nodes. The most important components are the *kube–apiserver* that exposes an API to interact with the cluster, *etcd* as distributed persistence layer to keep track of the cluster data, the *kube–scheduler* which is assigning pods to available nodes based on a set of policies, and the *kube–controller–manager* that is in charge of managing the lifecycle of a node and to expose service endpoints. Each node runs a *kubelet* that ensures the execution of containers in a pod and a *kube–proxy* to realize networking between nodes.[6]

K8s can be installed via provided package repositories. Using K8s involves the startup of the control plane, adding worker nodes to the cluster, and applying a deployment (a description of the workload to be executed). The same steps need to be done vice versa to tear down the cluster completely. At this, K8s requires at least 2 vCPUs with 2 GB memory.[7]

MicroK8s. Maintained by Canonical, mK8s aims to simplify the usage of K8s on public and private clouds by providing a lightweight and fully compliant K8s distribution, especially for low–end application areas like IoT.[8] By default, mK8s enables all basic components of K8s (like api–server, scheduler, or controller–manager) to make the cluster available. Further add–ons (e.g., DNS, ingress, or the metrics–server) can be enabled with one single command.[9] The realiza-

[6] https://kubernetes.io/docs/concepts/overview/components/
[7] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
[8] https://microk8s.io/docs
[9] https://microk8s.io/docs/addons

tion of high–availability, where multiple nodes carry the control plane and the datastore, can be achieved with a few commands.[10]

mK8s is provided by *snap*, Canonicals's package manager that runs applications in a sandbox.[11] Instead of etcd, which is used by K8s, mK8s uses *Dqlite* as high–availability datastore.[10,12] It is recommended to run mK8s with at least 4 GB of memory and 20 GB of storage (SSD recommended).[8]

K3s. Rancher offers K3s as lightweight K8s distribution, also with focus on low–end application areas. It is also fully compliant to K8s, contains all basic components by default, and targets a fast, simple, and efficient way to provide a highly available and fault–tolerant cluster to a set of nodes. The deployment takes place via one single and small binary including dependencies.[3]

Similar to mK8s, Rancher replaced etcd by another datastore, here *sqlite3*. Also, in–tree storage driver and cloud provider components are removed to keep the size small. K3s tries to lower the memory footprint by a reorganization of the control plane components of the cluster. The K3s master and workers, also called server and agents, encapsulate all the components in one single process.[13]

K3s is installed via a shell script that allows it to be run as a server or agent node. To achieve high–availability, new worker nodes can be easily added to the cluster by running a few commands.[14] The minimum hardware requirements are at least 1 vCPU and 512 MB of memory.[15]

4 Performance Comparison

This chapter covers the performance comparison of the three considered platforms. In the first place, we describe our experimental approach and environment shortly. Afterward, we analyze our obtained results from the experiment.

4.1 Experimental Setup and Design

To evaluate the resource consumption, we set up a controlled environment to achieve reproducible, comprehensible, and consistent results. According to the recommended system requirements (Section 3), we used four Ubuntu 20.04 Virtual Machines (VMs) with 2 vCPUs, 4 GB memory and a fast SSD with a capacity of 50 GB each. All VMs run on-premises on one single physical host machine with Kernel–based Virtual Machine (KVM) as hypervisor and *containerd* as container runtime. The host machine is equipped with a AMD Ryzen 7 3700X CPU (16 cores), 64 GB memory and a fast SSD. We deployed *netdata* as monitoring tool on all machines to collect data about the system utilization with a

[10] https://microk8s.io/docs/high-availability
[11] https://snapcraft.io/docs/getting-started
[12] https://kubernetes.io/docs/concepts/overview/components/
[13] https://rancher.com/docs/k3s/latest/en/architecture/
[14] https://rancher.com/docs/k3s/latest/en/
[15] https://rancher.com/docs/k3s/latest/en/installation/installation-requirements/

sample rate of 5 s. The collected data is stored in a document–oriented database on another machine. Netdata's monitoring agent creates a small CPU utilization of around 1 %, a negligible memory usage and disk utilization.[16]

In order to evaluate the entire lifecycle, we extended the approach of Fathoni et al. [2] as follows: We redefined the set of events to be evaluated. That means, we collect the CPU, memory, and I/O utilization during starting, adding, running, draining, and stopping of nodes, as well as applying, running, and deleting a small example deployment (nginx–deployment) with three replicas.[17]

The experiment is structured as follows: First, we installed all platforms with one master and three workers on the respective machines to create a high–availability cluster. Secondly, we stopped all platforms and platform–related services to put the system into idle condition. Finally, predefined *ansible playbooks* instruct the machines to perform the different steps of the lifecycle model. All playbooks and further details of the implementation are available on Github[18]. In total, we performed 25 runs per platform and averaged the results. The raw data and detailed metrics for all runs are available on GitHub[19] as well.

4.2 Experimental Results

Figure 1 shows the average resource utilization by master and worker nodes for all platforms over time. The small numbers at the top of each diagram imply the middle of the different steps in the lifecycle simulation.

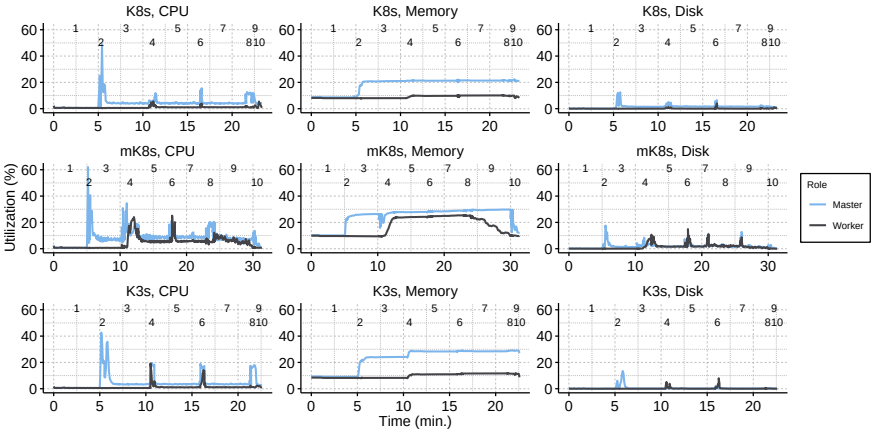


Fig. 1: Lifecycle analysis for K8s, MicroK8s, and K3s.

[16] https://github.com/netdata/netdata#features
[17] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[18] https://github.com/spboehm/kns-profiling
[19] https://spboehm.github.io/kns-profiling/

Table 1: Average resource consumption (μ/σ) for all platforms and events.

No.	Event	Role	K8s			mK8s			K3s		
			CPU	Memory	Disk	CPU	Memory	Disk	CPU	Memory	Disk
1	System idle	Master	0.64/0.3	8.66/0.81	0.05/0.09	0.76/0.35	10.16/1.12	0.08/0.11	0.6/0.27	9.15/0.8	0.04/0.09
		Worker	0.62/0.29	8.04/0.59	0.04/0.09	0.74/0.3	9.69/0.81	0.07/0.12	0.62/0.25	8.28/0.51	0.04/0.08
2	Start master	Master	19.15/17.87	13.32/3.98	4.15/5.55	24.78/20.43	20.6/4.64	4.88/7.61	28.83/12.65	16.46/4.71	2.68/2.84
		Worker	0.5/0.15	7.99/0.58	0.01/0.03	0.61/0.17	9.6/0.79	0.04/0.07	0.52/0.14	8.22/0.5	0.01/0.05
3	Master idle	Master	4.67/2.82	20.83/0.77	1.87/1.92	8.61/6.08	25.88/1.54	2.19/3.15	5.73/8.28	23.82/1.11	1.12/3.44
		Worker	0.57/0.16	8/0.56	0.02/0.06	0.66/0.18	9.49/0.76	0.03/0.07	0.57/0.16	8.24/0.48	0.02/0.09
4	Add workers	Master	6.33/2.59	21.27/0.65	3.18/1.77	15.95/10.62	25.91/3.13	3.23/2.82	14.99/6.88	27.07/2.08	0.39/0.37
		Worker	2.77/3.51	9.06/0.96	0.57/1.51	12.67/19.76	16.02/6.36	3.63/6.07	8.37/10.84	9.66/1.03	1.93/3.66
5	Cluster idle	Master	4.27/1.12	21.3/0.64	1.71/0.46	8.83/2.58	27.85/1.1	1.87/0.83	3.77/2.62	28.38/0.99	0.23/0.22
		Worker	1.1/0.4	9.77/0.48	0.05/0.18	5.78/3.49	23.99/1.14	1.92/2.13	1.26/0.86	10.95/0.43	0.16/0.67
6	Apply deployment	Master	7.19/4.53	21.3/0.61	3.12/1.92	17.72/5.06	28.29/1.09	3.74/3.39	15.37/5.85	28.74/0.96	0.77/0.97
		Worker	1.67/1.02	9.73/0.51	0.49/1.02	11.03/7.72	24.42/1.04	4.41/3.78	5.63/6.38	11.18/0.44	1.59/2.79
7	Deployment idle	Master	4.23/1.33	21.4/0.62	1.67/0.63	9.01/2.5	28.6/1.07	2.46/2.82	3.69/2.51	28.48/0.99	0.23/0.2
		Worker	1.16/0.38	10.06/0.43	0.07/0.28	5.92/2.77	25.03/0.99	2.49/3.26	1.38/2.2	11.61/0.4	0.18/1.4
8	Delete deployment	Master	10.71/2.96	21.5/0.6	2/0.57	17.63/3.63	29.18/0.98	2.34/1.32	14.67/6.03	28.82/1.07	0.31/0.15
		Worker	1.45/0.81	9.97/0.42	0.33/0.39	5.49/1.79	25.16/0.92	2.15/1.41	1.54/0.8	11.58/0.36	0.29/0.37
9	Drain workers	Master	4.58/1.69	21.71/0.5	1.36/0.16	7.77/2.52	29.55/1.03	1.89/2.78	3.56/2.26	28.89/0.73	0.28/0.2
		Worker	2/1.97	9.94/0.42	0.22/0.51	5.95/10.27	17.3/7.1	1.52/3.24	2.08/1.5	11.37/0.47	0.08/0.1
10	Stop master	Master	3.91/0.65	21.53/0.7	1.24/0.17	3.38/3.91	15.61/6.12	0.66/1.48	3.15/0.87	27.16/1.46	0.24/0.11
		Worker	0.64/0.14	8.31/0.48	0.1/0.08	0.71/0.21	9.89/0.66	0.07/0.12	0.65/0.18	8.71/0.29	0.18/0.29

For instance, event no. 2 involves the event *Start master*. This explains the very short peek in the CPU and I/O utilization as well as the increasing amount of memory. To measure the utilization of all idle events (i.e., event no. 1, 3, 5, and 7 - see Table 1 for details), we defined a time interval of five minutes to obtain the average utilization. Apart from mK8s, K8s and K3s are showing similar results, especially in terms of the time needed to pass through all steps in the experiment. Only mK8s needed more time ($\mu = 1860\,\text{s}$ / $\sigma = 12.9\,\text{s}$) in comparison to K8s (1379 s / 17.1 s) and K3s (1361 s / 9.47 s). The platforms K8s and K3s cause similar load profiles for all metrics as displayed in Table 1. However, K3s burdens CPU and memory slightly more than K8s but shows a smaller average and volatility in disk utilization. K3s has shown slightly better results for CPU and disk utilization in comparison to K8s only for a few events (e.g., event no. 5, 7, 9, and 10). The memory consumption of mK8s and K3s is quite similar. However, mK8s shows the highest utilization for CPU and disk. On average, the master nodes mostly need more resources compared to the worker nodes because cluster–managing services are located there.

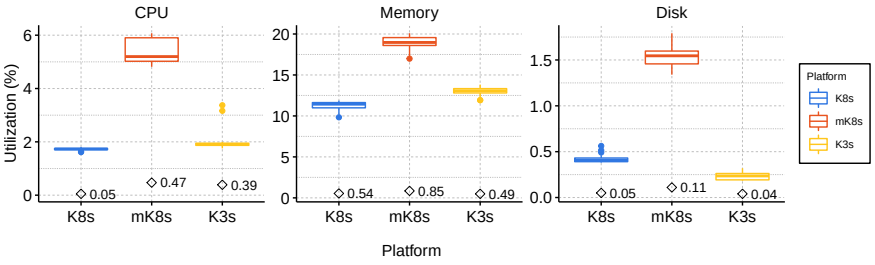


Fig. 2: Resource utilization for K8s, MicroK8s, and K3s ($\diamond = \sigma$).

Figure 2 shows the average utilization of all events for all platforms as box plots. Furthermore, the standard deviation is displayed ($\diamond = \sigma$). The previously described results are also reflected in this overall and averaged comparison. The platform mK8s shows the worst results for all metrics. Regarding CPU and memory utilization, the differences between K8s and K3s are very small. K3s shows a slightly smaller disk utilization. Consequently, the claim of lightweight K8s holds only partially. All platforms exhibit a small σ which indicates that the experiment created stable results under repeated simulations. Also, the box plots do neither show an extensive amount of outliers nor a high dispersion.

The taken amount of time for selected steps in the cluster lifecycle is displayed in Figure 3. The values refer to all four nodes. K3s shows better results for nearly all events compared to K8s, except the needed time to apply deployments and drain workers. In this comparison, mK8s is quite close to K8s but needs a very long time for adding and draining workers. mK8s starts a master node a bit faster compared to K8s but slower than K3s. The creation of new deployments happens nearly in the same time for mK8s and K3s. K8s applies the deployment around four times faster than the other platforms. The deletion of a deployment by mK8s roughly takes twice the time K8s or K3s needs. Tearing down the cluster happens rapidly as well, mK8s needs around 32 seconds.

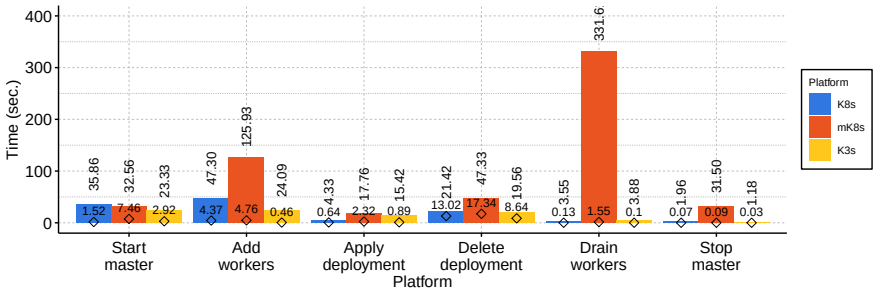


Fig. 3: Average time consumption for K8s, MicroK8s, and K3s ($\diamond = \sigma$).

5 Discussion

The obtained results from the previous chapter can be explained with the different characteristics of the lightweight platforms (Section 3). K3s has shown a very small I/O utilization potentially due to the usage of sqlite3 instead of etcd as database. In terms of time, K3s shows merits for all events except applying deployments or draining workers compared to K8s. Bundling all components of K8s into one single process may lead to this performance enhancement. mK8s had overall a higher resource utilization. Especially adding and draining workers needed a long time. The reason for this is that mK8s is optimized for a

single-node cluster. When a particular node leaves the cluster, it restarts again as single-node K8s automatically.[20] There may be an option to avoid this restart and reduce the time for draining nodes by stopping mK8s forcefully. However, we followed the official documentation, which also states that adding and graceful draining of nodes may require minutes.[10] It is worth noting that mK8s has the highest system requirements, followed by K8s and finally K3s (Section 3).

The proposed experiment and the obtained results underlie a few limitations. Firstly, we tested all platforms running on a virtualized setup with KVM and containerd. Other alternatives may have an impact on the experimental results. However, containerd is the recommended runtime for mK8s and K3s.[21,22] We measured only the utilization on the overall system–level without network utilization and did not consider the utilization of K8s–related processes in detail. Furthermore, we modeled the entire lifecycle with a limited set of machines to obtain results on how long each platform needs to perform a set of actions.w We did not burden the nginx deployment and kept everything in idle condition. However, a performance comparison based on applications was not in focus of our research. There may be a need for synthetic benchmarks for lightweight on–premises container platforms, such as performed by Ferreira and Sinnott [3].

6 Conclusion and Future Work

This paper showed an approach to compare different K8s distributions in a quantitative way. To answer our research question, we conclude that replacing K8s with a lightweight distribution can be beneficial in particular circumstances. For the most part, there are only small differences regarding the resource utilization between K8s and K3s. mK8s showed a higher resource and time consumption for nearly all events. K3s has shown a better performance except applying deployments and draining workers in terms of needed time. Although our experiment shows only preliminary results and the findings are limited to some extent, we argue that not all platforms fulfill the claim being more lightweight compared to K8s. Particularly areas like Fog, Edge, and IoT computing with a highly varying number of nodes over time can benefit from lightweight K8s platforms.

We plan to enrich our proposed simulation model with detailed analysis at the process–level for future work. To increase the overall validity, we want to run the experiment at a larger scale to get better insights how the different distributions manage a larger number of nodes and workloads. Furthermore, we want to deepen the statistical analysis to obtain significant differences between the platforms regarding resource utilization and time consumption of our lifecycle model. Finally, it is worth considering the qualitative dimension. As pointed out in Section 3, each platform provides various ways to deploy and interact with the cluster. We want to consistently evaluate these differences in a qualitative survey by taking other lightweight platforms like KE and MK into consideration.

[20] https://microk8s.io/docs/clustering
[21] https://microk8s.io/docs/configuring-services
[22] https://rancher.com/docs/k3s/latest/en/advanced/

References

1. Eiermann, A., Renner, M., Großmann, M., Krieger, U.R.: On a fog computing platform built on ARM architectures by docker container technology. In: Innovations for Community Services, pp. 71–86. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-60447-3_6
2. Fathoni, H., Yang, C.T., Chang, C.H., Huang, C.Y.: Performance comparison of lightweight kubernetes in edge devices. In: Pervasive Systems, Algorithms and Networks, pp. 304–309. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-30143-9_25
3. Ferreira, A.P., Sinnott, R.: A performance evaluation of containers running on managed kubernetes services. In: 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 199–208. IEEE (2019). https://doi.org/10.1109/cloudcom.2019.00038
4. Goethals, T., Turck, F.D., Volckaert, B.: FLEDGE: Kubernetes compatible container orchestration on low-resource edge devices. In: Internet of Vehicles. Technologies and Services Toward Smart Cities, pp. 174–189. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-38651-1_16
5. Javed, A., Heljanko, K., Buda, A., Framling, K.: CEFIoT: A fault-tolerant IoT architecture for edge and cloud. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 813–818. IEEE (2018). https://doi.org/10.1109/wf-iot.2018.8355149
6. Kayal, P.: Kubernetes in fog computing: Feasibility demonstration, limitations and improvement scope : Invited paper. In: 2020 IEEE 6th World Forum on Internet of Things, pp. 1–6. IEEE (2020). https://doi.org/10.1109/wf-iot48130.2020.9221340
7. Kristiani, E., Yang, C.T., Huang, C.Y., Wang, Y.T., Ko, P.C.: The implementation of a cloud-edge computing architecture using OpenStack and kubernetes for air quality monitoring application pp. 1–23 (2020). https://doi.org/10.1007/s11036-020-01620-5
8. Medel, V., Rana, O., ángel Bañares, J., Arronategui, U.: Modelling performance & resource management in kubernetes. In: Proceedings of the 9th International Conference on Utility and Cloud Computing, pp. 257–262. ACM (2016). https://doi.org/10.1145/2996890.3007869
9. Medel, V., Tolosana-Calasanz, R., Bañares, J.Á., Arronategui, U., Rana, O.F.: Characterising resource management performance in kubernetes. vol. 68, pp. 286–297. Elsevier BV (2018). https://doi.org/10.1016/j.compeleceng.2018.03.041

All links were last followed on February 20, 2021.

An Evaluation of Saga Pattern Implementation Technologies

Karolin Dürr, Robin Lichtenthäler, and Guido Wirtz

Distributed Systems Group, University of Bamberg

Abstract. The Saga pattern is frequently mentioned in the literature to structure communication workflows in Microservices Architectures. To ease the implementation of the Saga pattern frameworks and tools have emerged. By implementing an exemplary use case, we qualitatively evaluate two of such technological solutions in this paper according to criteria relevant for Microservices Architectures. This evaluation can be considered when deciding on which technology to use for implementing the Saga pattern, or also as a more general insight into what should be kept in mind when implementing the Saga pattern in Microservices Architectures.

Keywords: Microservices, Saga pattern, Workflow

1 Introduction

The *Microservices Architecture* is a pattern that emerged from real-world usage and constitutes a fast-moving topic [5, 9, 16]. A microservice is a small and autonomous service modeled around a business domain. A distributed system that consists of numerous microservices represents the Microservices Architecture [5] where data storage is ideally not shared, but owned exclusively by each microservice. Communication between microservices happens via messages over the network [9]. The main advantage is service independence enabling independent deployability, maintainability and evolvability of services [11]. A main challenge, however, is inter-service communication, because communication over the network is comparatively slow and unreliable [5]. The question arises how this communication can be structured and managed, especially for complex business scenarios with multiple services which even require certain transactional guarantees [16].

Such a complex business scenario would be booking a trip where the booking includes several steps such as booking a flight, a hotel, and a rental car. Applied to a Microservices Architecture where different microservices are responsible for the different steps, this scenario has also been used by Catie McCaffrey in a conference talk[1] to motivate the usage of the Saga pattern.

Because all steps are required to book a trip as a whole, a classical approach would be to use a distributed transaction for example with the *2-Phase Commit*

[1] `https://www.youtube.com/watch?v=xDuwrtwYHu8`, last accessed: 2021-02-17

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at `http://ceur-ws.org/Vol-2XXX`

(2PC) protocol [1, 9]. However, classical distributed transactions contradict the service independence characteristic of a Microservices Architecture. First, the 2PC protocol depends on the availability of all participants [7, 11]. If one participant fails, the system as a whole becomes unavailable. Second, the scalability is affected, because 2PC participants need to lock resources which can affect the overall transactional throughput and lead to competitive situations [9, 12]. And third, distributed transactions are missing support from modern technologies, like NoSQL databases or message brokers [11].

Therefore, other approaches have been discussed [3, 7, 9] with the Saga pattern being mentioned frequently [6, 10, 11, 15]. The Saga pattern divides a transaction that might take a long time into multiple local ones. Thereby, it reduces the dependence on the availability of all participants at the same time and prevents the need to lock all included resources until full completion. Although it can therefore not provide the same transactional guarantees as, for example, the 2PC protocol, it aligns better with the characteristics of Microservices Architectures. Using the Saga pattern for the trip booking scenario means that still the whole trip booking needs to be supervised by one service. However, the included steps, such as booking a hotel or booking a flight, are done more independently in local transactions by the different services involved.

Because this separation into multiple more independent transactions leads to additional challenges, implementing the Saga pattern can get complex. Therefore framework support is desirable and some technological solutions have emerged. The goal of this paper is to investigate the capabilities offered by existing frameworks with a focus on orchestrated Sagas and the context of characteristics and challenges of Microservices Architectures. This is summarized in the following research question:

> **RQ:** How well do recent technological solutions support implementing the Saga pattern concerning the design, the execution and the visualization of communication between microservices?

In Sect. 2, our approach to answer the research question is described. In Sect. 3, the details of the Saga pattern are depicted based on literature and the already introduced example scenario. This is used as a foundation for the following evaluation in Sect. 4, our main contribution. Finally, we draw a conclusion in Sect. 5.

2 Methodology

First, we carried out a literature review to understand the Saga pattern itself and its applicability to the Microservices Architecture. As sources, we considered the original paper for the Saga pattern [6], as well as more recent books [3, 10, 11] and papers [8, 15] which add the context of microservices. The result is the description of the Saga pattern in Sect. 3 based on the trip booking example.

We then used this example to implement the Saga pattern with available technological solutions. Solutions that have emerged so far are Axon[2], Eventuate Tram[3], Netflix Conductor[4], and more recently Long Running Actions for MicroProfile[5]. However, because this work has been done as a part of the bachelor thesis of the first author, we had to limit the scope and therefore only selected two solutions. The first solution we selected for our evaluation is Eventuate Tram, because it is specifically designed for the Saga pattern and described in detail in [11]. And the second solution is Netflix Conductor, because Netflix as a company is well-known for its successful microservices approach [2]. Furthermore, Netflix Conductor was not considered in another similar study [15] which compared technological solutions for the Saga pattern. The study by Štefanko et al. [15] includes a small set of criteria for comparing the different solutions which are not explained in detail in the paper and additionally discusses problems of the solutions in a qualitative way. Furthermore, Štefanko et al. [15] conducted a performance test to measure processing times and throughput. In contrast, we derived a more comprehensive criteria catalog from general Saga execution characteristics as well as from considering Microservices Architecture characteristics and challenges to evaluate the solutions. Our evaluation is qualitative, because we assess the solutions according to the criteria catalog based on our implementations. We have not performed quantitative evaluations, like performance benchmarks to assess scalability and throughput or user experiments to assess the ease of use. With respect to the work of Štefanko et al. [15], our work extends it by evaluating additional criteria and considering an additional solution. The resulting evaluation of Eventuate Tram and Netflix Conductor based on these criteria is presented in Sect. 4.

3 The Saga Pattern

The Saga pattern was introduced by Garcia-Molina and Salem [6] for long lived transactions by designing them as a sequence of local transactions. Although they focused on a centralized system, they also mentioned the possibility of a distributed implementation [6]. Therefore, Sagas have been proposed for updating

[2] `https://docs.axoniq.io/reference-guide/axon-framework/sagas`, last accessed 2021-02-17

[3] `https://eventuate.io/`, last accessed 2021-02-17

[4] `https://netflix.github.io/conductor/`, last accessed 2021-02-17

[5] `https://microprofile.io/project/eclipse/microprofile-lra`, last accessed 2021-02-17

data in multiple services in a Microservices Architecture without using distributed transactions [11].

To clarify this, Fig. 1 shows an exemplary execution of the trip booking example. The system offers the possibility to book a trip which includes booking a hotel and a flight. This can be considered as a long lived transaction with three microservices involved: a *Travel Service* which accepts requests for booking a trip and initiates the execution, a *Hotel Service* which manages hotel bookings, and a *Flight Service* which manages flight bookings. Using the 2PC protocol would mean that booking the hotel and the flight would be done within one ACID [14] transaction coordinated by the Travel Service where the whole trip is either booked or rejected. During the transaction execution, all services must lock resources impacting throughput [9, 12]. If one service is temporarily unavailable, the transaction may fail reducing the availability of the system as a whole [7, 11].

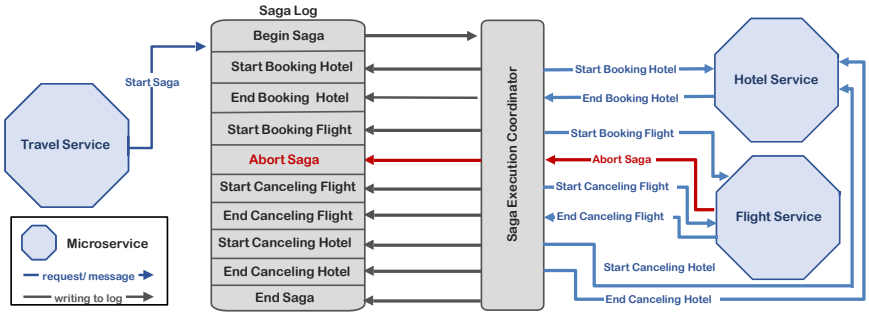


Fig. 1. Execution of a Saga's failure scenario based on [6].

Implementing the example as a Saga means that the long lived transaction is split into three local transactions: Save Trip Information, Booking Hotel, and Booking Flight. The Save Trip Information transaction is executed upon a trip booking request locally by the Travel Service to initiate the Saga and ensure Durability. It is part of the Begin Saga step and does not require communication with another service which is why it is not explicitly shown in Fig. 1. The Booking Hotel transaction and the Booking Flight transaction are executed locally in the Hotel Service and Flight Service, respectively. Each local transaction updates only the data within one service and then triggers the next one [8, 11] until all transactions are completed, and hence the Saga itself completes. If one transaction fails, the Saga aborts and all previously completed transactions have to be compensated. This case is shown in Fig. 1, where after the booking of a flight fails, the previously done hotel booking has to be canceled. Consequently,

[6] `https://speakerdeck.com/caitiem20/applying-the-saga-pattern?slide=70`, slide 70, last accessed: 2021-02-17

for each local transaction, a compensating transaction also needs to be provided, which can compensate the transaction completely or at least semantically [6, 10].

In contrast to the 2PC protocol, a service therefore only holds locks for local transactions and not for the whole Saga execution enabling it to effectively serve more requests. One consequence is that the Isolation property is not satisfied because intermediate results are visible to other Sagas before the executing one is fully committed [15]. Therefore, countermeasures need to be taken to prevent anomalies resulting from the lack of Isolation [11]. Also, Atomicity is not given for the Saga as a whole, solely for each local transaction [6, 10]. Instead of strict Consistency, only eventual consistency [13] is provided [10, 15]. During execution, a trip with a hotel, but no flight would be inconsistent, but after completion consistency is again achieved, when necessary through compensations. Durability is fully guaranteed through the durability of local transactions and the Saga log, which is a distributed log to persist every executed transaction. The Saga log is managed by a component called the Saga Execution Coordinator which is itself stateless and uses the log to trigger transactions and thereby proceed Saga executions [6]. Having a Saga Execution Coordinator either as a separate service or within a service exemplifies the orchestrated Saga approach with the Saga Execution Coordinator being called the orchestrator [3, 10]. Although out of the scope of this work, also a choreographed approach would be possible where the coordination is distributed [10].

4 Technological Evaluation

Before we discuss the evaluation based on a set of criteria, some fundamental differences need to be mentioned, because they also affect our evaluation results. Eventuate Tram specifically focuses on Sagas by offering a Java-based Domain Specific Language (DSL) for specifying a sequence of transactions and corresponding compensating transactions inside the service acting as the Saga orchestrator. It is then executed together with the so-called CDC service and infrastructure components such as a database for persisting the Saga log and a message broker for communication. The DSL can also be used for the participants, if implemented with Java. For other languages, participants have to be integrated based on the used communication mechanisms. We implemented all services as Spring[7] services with the DSL included. In contrast, Conductor is not designed explicitly for Sagas, but distributed workflows in general. The central component is the Conductor server which accepts workflows in the form of a JSON-based DSL. A Saga is registered as a workflow, with tasks representing transactions for which different types are offered. We used so-called worker tasks which are more customizable than others. They need to be registered, and the services, again implemented in Java, can then poll and update these tasks to proceed with the workflow. All implementations with examples and detailed information on execution can be found online[8].

[7] `https://spring.io/`, last accessed 2021-02-17

[8] `https://github.com/KarolinDuerr/BA-SagaPattern`

Table 1. Evaluation overview

Criterion	Eventuate Tram	Netflix Conductor
General Saga Characteristics		
Specifying compensating transactions (CT)	✓	✓
Automated execution of CTs	✓	✓
Compensation only where needed	✓	not directly supported
Parallel execution of transactions	✗	✓
Choreographed Sagas	✓	✗
Monitoring		
Runtime state of Sagas	via database	UI visualization
Orchestrator metrics	from CDC service	from Conductor server
Tracing	Zipkin integration	not directly supported
Logging	microservices logs	Conductor server logs
Expandability		
Relatively simple integration	✓	✓
Terminating or pausing running Sagas	not directly	via UI
Versioning Sagas	✗	✓
Built-in language support	Java	Java, Python
Any language for orchestrator	✗	✓
Any language for participant	✓	✓
Failure performance		
Enforced execution timeouts	✗	✓
Retry of failing participant without restart	✗	✓
Independent compensating transactions	✓	✗
Auto-continuation after orchestrator crash	✗	✓
No. of services for orchestration	2	1
New Sagas while orchestrator unavailable	✓	only with buffering
High availability	through replication	through replication

Our first set of criteria (see Table 1 for an overview of all results) covers *general characteristics*. Both technologies allow for specifying compensating transactions which are also automatically triggered. However, only Eventuate allows for mapping compensating transactions to transactions so that only needed compensating transactions are executed while Conductor allows for one failure workflow per workflow. That means the failure workflow must contain all compensating transactions and even compensating transactions which would not have been necessary are executed in case of a Saga abort. This is because Conductor is not specifically focused on Sagas. With Conductor, the central component is the Conductor server which orchestrates the Saga execution, and participants are connected to the Conductor server, which is why it does not support a choreographed approach to Sagas. With Eventuate as a framework however, the participants could also be connected directly with each other, enabling also a choreographed approach. In contrast, transaction execution in Eventuate is strictly sequentially, while Conductor also allows for parallel execution of transactions.

The second set of criteria considers *monitoring*, a challenge in Microservices Architectures [2, 4]. To get insights at runtime, Eventuate offers no pre-built tool, but the database tracking all transactions and messages could be used as a source for building a custom monitoring solution. Instead, Conductor offers a UI which visualizes current workflows and provides useful functionalities for runtime insights. A metrics endpoint exists for both technologies, which can be used to collect metrics like the number of sent messages, average execution times, or the number of failed Saga workflows. A possibility to use distributed tracing is only given by Eventuate which offers a pre-built Zipkin[9] integration. Additionally, logs are written by both technologies which could help with troubleshooting.

Because Microservices Architecture-based systems change and evolve, the third set of criteria covers *expandability*. We extended the example with an additional service, which can also be found in the repository. For both technologies, the integration was possible without significant problems. Nevertheless, Conductor is suited better for updating or extending a running system because currently executing Sagas can be managed via the UI and workflows can be versioned. This means that Sagas of a new version can be started at the same time as there are still Sagas of an old version executing. With Eventuate, handling updates at runtime requires more effort. Regarding polyglot programming as a characteristic of Microservices Architectures [9, 16], Eventuate is a bit more restricted because the DSL is based on Java which means that the orchestrator also needs to be written in Java. With Conductor, the Conductor server is mainly responsible for the orchestration which means that the service starting a Saga can be written in any language. In addition to a pre-built Java client for writing participants, Conductor also offers a Python client.

As a final set of criteria, we consider the technologies' *handling of failures* which have to be expected in a distributed system. Both technologies tolerate possible crashes of Saga participants by retrying communications. However, only

[9] `https://zipkin.io/`, last accessed 2021-02-17

Conductor enforces an execution timeout to be set while Eventuate might, per default, wait indefinitely for a service to restart. Depending on the use case and volume of requests, this can become an issue. If there is no execution timeout for Sagas, a service being unavailable for an extended period together with a high volume of requests might lead to an overloaded system as a whole, because Saga executions pile up and cannot make progress. An execution timeout can then protect the system from consequential failures. Then again, there might be use cases where Sagas should not be stopped at all because of a timeout. In such a case, the enforced execution timeout of Conductor might be problematic. A participant responding with a failure is unsubscribed from the message broker with Eventuate, requiring a full restart of the participant so that it can re-register. In contrast, Conductor retries even if a participant responded with a failure that might only be temporary. Because compensating transactions are executed only where needed with Eventuate, they can be executed independently from participants where no compensation is necessary. Thus, also crashes of such participants can be tolerated. Crashes of the Saga coordinator are tolerated by both technologies and execution can continue afterwards because all necessary information is persistently logged. However, merely Conductor automatically continues while Eventuate needs a trigger after restart, such as a new Saga start. With Eventuate, two services are required for orchestration: The CDC service as orchestrator and a service in control of the Saga. Therefore, new Sagas can still be started if only the orchestrator is unavailable. In case of Conductor, the Conductor server is the exclusive orchestrator and additional logic would be needed to buffer new requests in another service. Finally, both can be set up as a highly available system by replicating the CDC service or the Conductor server, respectively.

To summarize, both technologies enable robust Saga implementations. The characteristics of the Saga pattern are represented more clearly with Eventuate than with Conductor. However, Eventuate comes with limitations regarding the flexibility in operation which is in turn better supported by Conductor.

5 Conclusion and Outlook

Given the Microservices Architecture as a popular software architecture approach, patterns and technologies are needed to efficiently implement these systems and tackle their accompanying challenges. Our evaluation of Saga pattern implementation technologies covers one of the aspects software engineers should consider to make an informed decision on which technologies to use based on their specific needs. As future work, we want to include additional technologies into our evaluation, such as Axon and Long Running Actions for MicroProfile, but also the possible usage of BPMN workflow engines as proposed in a recent talk by Bernd Rücker[10] and also by Niall Deehan at the ZEUS 2020 workshop. Furthermore, extending the evaluation with quantitative methods is imaginable, for example by doing a performance benchmark.

[10] `https://www.youtube.com/watch?v=7uvK4WInq6k`, last accessed: 2021-02-17

References

1. Al-Houmailya, Y.J., Samaras, G.: Two-Phase Commit. In: Encyclopedia of Database Systems, pp. 3204–3209. Springer US (2009), https://dx.doi.org/10.1007/978-0-387-39940-9_713
2. Alshuqayran, N., Ali, N., Evans, R.: A Systematic Mapping Study in Microservice Architecture. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). pp. 44–51. IEEE Computer Society (2016), https://dx.doi.org/10.1109/SOCA.2016.15
3. Bruce, M., Pereira, P.A.: Microservices in Action. Manning Publications, 1st edn. (2018), ISBN: 9781617294457
4. Cerny, T., Donahoo, M.J., Trnka, M.: Contextual Understanding of Microservice Architecture: Current and Future Directions. ACM SIGAPP Applied Computing Review 17(4), 29–45 (2018), https://dx.doi.org/10.1145/3183628.3183631
5. Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: Yesterday, Today, and Tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216. Springer International Publishing (2017), https://dx.doi.org/10.1007/978-3-319-67425-4_12
6. Garcia-Molina, H., Salem, K.: Sagas. In: Proceedings of the 1987 Association for Computing Machinery Special Interest Group on Management of Data (ACM SIGMOD) International Conference on Management of Data. vol. 16, pp. 249–259. ACM Press (1987), https://dx.doi.org/10.1145/38714.38742
7. Helland, P.: Life Beyond Distributed Transactions: An Apostate's Opinion. ACM Queue 14(5), 69–98 (2016), https://dx.doi.org/10.1145/3012426.3025012
8. Limón, X., Guerra-Hernández, A., Sánchez-García, A.J., Arriaga, J.C.P.: SagaMAS: A Software Framework for Distributed Transactions in the Microservice Architecture. In: 2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT). pp. 50–58. IEEE Computer Society (2018), https://dx.doi.org/10.1109/CONISOFT.2018.8645853
9. Newman, S.: Building Microservices - Designing Fine-Grained Systems. O'Reilly Media, Inc., 1st edn. (2015), ISBN: 9781491950357
10. Newman, S.: Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, Inc., 1st edn. (2019), ISBN: 9781492047841
11. Richardson, C.: Microservices Patterns. Manning Publications, 1 edn. (2019), ISBN: 9781617294549
12. Thomson, A., Diamond, T., Weng, S.C., Ren, K., Shao, P., Abadi, D.J.: Calvin: Fast Distributed Transactions for Partitioned Database Systems. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 1–12. Association for Computing Machinery (2012), https://dx.doi.org/10.1145/2213836.2213838
13. Vogels, W.: Eventually Consistent. Communications of the ACM 52(1), 40–44 (2009), https://dx.doi.org/10.1145/1435417.1435432
14. Vossen, G.: ACID Properties. In: Encyclopedia of Database Systems, pp. 19–21. Springer US (2009), https://dx.doi.org/10.1007/978-0-387-39940-9_831
15. Štefanko, M., Chaloupka, O., Rossi, B.: The Saga Pattern in a Reactive Microservices Environment. In: Proceedings of the 14th International Conference on Software Technologies (ICSOFT) 2019. pp. 483–490. SciTePress (2019), https://dx.doi.org/10.5220/0007918704830490
16. Zimmermann, O.: Microservices Tenets. Computer Science - Research and Development 32(3-4), 301–310 (2016), https://dx.doi.org/10.1007/s00450-016-0337-0

Systematic Literature Tools: Are we there yet?

Dominik Voigt[1,2], Oliver Kopp[1], and Karoline Wild[2]

[1] JabRef e. V.
Sindelfingen, Germany
[firstname]@jabref.org
[2] University of Stuttgart
Institute of Architecture of Application Systems
Stuttgart, Germany
karoline.wild@iaas.uni-stuttgart.de

Abstract. The number of publications is steadily growing. systematic literature reviews (SLRs) are one answer to this issue. A variety of tools exists designed to support the review process. This paper summarizes requirements for adequate tooling support and shows that existing tools do not meet all of them. We further investigate whether reference management tools can be used in conjunction with existing SLR tools to address the current gaps in supporting SLRs. For that we evaluate three reference management tools, JabRef, Bibsonomy, and Zotero, against currently unadressed requirements and outline the next steps.

Keywords: methods, systematic literature review, tool support

1 Introduction

With the ever-growing number of publications in computer science [15], and other fields of research, the conduction of meta studies becomes necessary to keep up [17, 20]. Kitchenham [10] introduced the *systematic literature review (SLR)* method to address this issue. The main idea is to systematically search and evaluate all existing publications regarding a specific topic.

Computer science researchers that conduct SLRs face three main challenges [18]: (i) For SLR novices, the learning of the SLR process and the definition of the research protocol is challenging. (ii) All SLR practitioners face difficulties assessing the quality of primary studies, a critical step within the conduction of an SLR, especially for qualitative studies. (iii) The access and acquisition of relevant studies across multiple e-libraries is a challenge. As a consequence, the need for appropriate tool support to address these challenges has been growing [1, 4, 8, 11, 18].

Currently, appropriate tool support is not yet achieved. To illustrate this, this paper (i) summarizes the requirements for adequate SLR tools, (ii) discusses the shortcomings of existing tools, and (iii) evaluates the capabilities of three reference management tools to investigate whether they can address these shortcomings. First, the overall SLR process and tool requirements are described in Sect. 2.

J. Manner, S. Haarmann, S. Kolb, N. Herzberg, O. Kopp (Eds.): 13th ZEUS Workshop, ZEUS 2021, Bamberg, held virtually due to Covid-19 pandemic, Germany, 25-26 February 2021, published at http://ceur-ws.org/Vol-2XXX

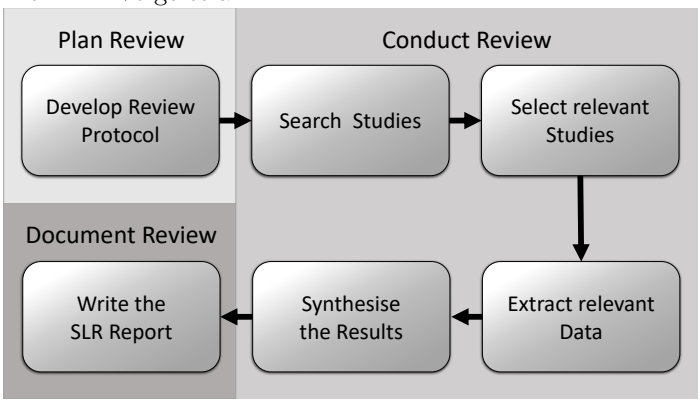


Fig. 1: Simplified SLR Process

Afterwards, we discuss in Sect. 3 that existing SLR tools do not fully meet these requirements yet. We claim that existing reference management tools can be extended and used in conjunction with existing SLR tools to overcome their current lack of support. Therefore, we evaluate three tools in Sect. 4. Finally, we conclude and outline the next steps required to close the remaining gaps in supporting SLRs in Sect. 5.

2 SLR Process and SLR Tool Requirements

The SLR process consists of three phases each with a set of steps [10]. These phases include the *planning* of the review, the *conduction* of the review, and the *reporting* of results. In Fig. 1 a simplified SLR process is depicted.

During the planning of the review, the review protocol is created, which especially describes the execution of the review. After the review is planned, it is executed during the conduction according to the defined review protocol: First the set of candidate papers is aggregated by executing the search strategy (*search step*). Then, the candidate filters are checked for inclusion using the selection criteria and quality instruments (*selection step*). Subsequently, the data is extracted (*extraction step*) and the results are synthesised (*synthesis step*). Finally, the results are reported in a format that makes them actionable and describes their significance.

The main challenge researchers face during the conduction of the SLR is during the search step. Commonly used e-libraries in the domain of computer science research, such as IEEE, arXiv, and ACM, do not support easy mass access, which is key to the SLR method as all relevant studies have to be found [1]. Furthermore, different e-libraries use different interfaces regarding their search syntax and capabilities [1]. Thus e-library specific search strings have to be crafted and papers have to be retrieved individually. This introduces a lot of unnecessary manual effort for the researcher [1, 4].

Thus, adequate tool support is required. To focus the development of tools, Al-Zubidy and Carver [1] identified 35 tool requirements based on interviews. The need for support during the search step becomes evident in the top 8 requirements (R1–R8, ranked by the number of survey respondents that mentioned the requirement): search multiple databases with a standardized query (R1, 48 times), removing duplicate studies (R2, 13 times), provide filtering for studies (R3, 7 times), merging new results into the existing database (R4, 6 times), synonym recommendation for search strings (R5, 6 times), a repository for studies (R6, 6 times), standardized export formats (R7, 6 times), automatic download of full-text papers (R8, 6 times). These requirements should be fulfilled by SLR tools to adequate support researchers during their review.

3 Available SLR Tools in Computer Science

Several SLR tools specific to computer science are available to support the overall SLR process [2, 3, 5, 6, 9, 16]. The tools have been evaluated and compared in several studies, most recently by Marshall et al. [14] and Al-Zubidy and Carver [1]. Both of these studies concluded that, while (i) the overall process support was at least partially sufficient, (ii) the support for the search and management of literature on the other hand was only partially supported at best. This is a significant downside as integrated search and study management are the most requested feature for SLR tools (R1–R8) [1, 7, 11, 17, 18].

Since the review of Al-Zubidy and Carver [1] we identified two new tools by using the SLR Toolbox [13]: CloudSERA [19] and Thot [12]. Both do not provide any significant improvement concerning these aspects.

Thus existing SLR tools have lacking support for the search and selection steps of the SLR process. To address this gap, we propose the use of reference management tools during the search and selection step, resulting in a conjunctive use with existing SLR tools to provide support for every step of the SLR process.

4 Evaluation of Reference Management Tools

There exists a variety of reference management tools that can potentially be used in conjunction with existing SLR tools. However, to allow adaptation to the needs of SLRs, we considered only open source reference management tools as candidates. Therefore, we extracted all open source reference management tools from this list [21]. Thus, we evaluate the JabRef[3], Bibsonomy[4], and Zotero[5] reference management tools. We evaluate them against the requirements enumerated in Sect. 2 (R1–R8). A summary of this evaluation is displayed in Table 1.

Zotero does not offer any way to search e-libraries. JabRef supports integrated search but requires separate search strings for different libraries (R1). The reason

[3] `https://www.jabref.org/`
[4] `https://www.bibsonomy.org/`
[5] `https://www.zotero.org/`

Table 1: Evaluation of candidate reference management tools

	R1	R2	R3	R4	R5	R6	R7	R8
Bibsonomy	✗	✓	✗	✓	✗	✓	✗	✓
JabRef	✓	✓	✓	✓	✗	✓	✓	✓
Zotero	✗	✓	✓	✓	✗	✓	✓	✓

for this is the different search syntax and capabilities offered by the commonly used e-libraries. Bibsonomy provides search over their own publication repository, but not over external e-libraries.

As reference managers, JabRef, Bibsonomy, and Zotero, can manage repositories of references, including removing duplicate studies, and merging new entries into their database (R2, R4, R6). Only JabRef and Zotero support filtering studies based on their metadata, Bibsonomy solely provides filtering based on the user defined tags (R3). Furthermore, only JabRef and Zotero allow the acquisition of full-text pdfs to their corresponding reference (R8). Neither JabRef, nor Bibsonomy, nor Zotero support the recommendation of synonyms for search strings (R5). All reference managers support the export of entries into commonly used formats, such as BibTeX, Endnote, and RIS (R7).

The reference management features and integrated search make JabRef the most promising candidate tool that can fill the support gap for the search and selection steps. Moreover, the standardized export formats it provides make the integration into any SLR tool chain quite simple. If JabRef can address the issue of requiring separate queries for the different e-libraries and the provision of synonyms for search string construction it could address all of its current shortcomings and fulfill all of the top 8 requirements.

5 Conclusion and Outlook

In this paper, we outlined the SLR tool landscape, requirements on them, and their current drawbacks. Open source reference management tools can fulfill some key requirements missing from existing SLR tools during the search and selection step of an SLR. Thereby, JabRef has the significant advantage over Zotero and Bibsonomy that it offers integrated search, which is also programmatically available. With the integrated search being the most demanded feature and JabRef has partial support for it, we plan to extend JabRef to address the problem of e-library specific query strings (R1). With this extension, the conjunctive use of JabRef with existing SLR tools will close the gap in support for the search and selection steps. This will enable computer science researchers crafting a solid basis of their related work search. Reducing the effort required for conducting SLRs and improving the overall quality of scientific research.

Acknowledgments This work was partially funded by the BMWi project *PlanQK* (01MK20005N).

References

1. Al-Zubidy, A., Carver, J.C.: Identification and prioritization of SLR search tool requirements: an SLR and a survey. Empirical Software Engineering 24(1), 139–169 (2018)
2. Barn, B.S., Raimondi, F., Athappian, L., Clark, T.: SLR-Tool: A tool to support collaborative systematic literature reviews. In: Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS). SCITEPRESS (2014)
3. Bowes, D., Hall, T., Beecham, S.: SLuRp. In: Proceedings of the 2nd international workshop on Evidential assessment of software technologies (EAST). ACM Press (2012)
4. Carver, J.C., Hassler, E., Hernandes, E., Kraft, N.A.: Identifying barriers to the systematic literature review process. In: International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE (2013)
5. Fabbri, S., Silva, C., Hernandes, E., Octaviano, F., Thommazo, A.D., Belgamo, A.: Improvements in the StArt tool to better support the systematic review process. In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE). ACM Press (2016)
6. Fernandez-Saez, A.M., Bocco, M.G., Romero, F.P.: SLR-TOOL – a tool for performing systematic literature reviews. In: 5th International Conference on Software and Data Technologies. SCITEPRESS (2010)
7. Hassler, E., Carver, J.C., Hale, D., Al-Zubidy, A.: Identification of SLR tool needs – results of a community workshop. Information and Software Technology 70, 122–129 (2016)
8. Hassler, E., Carver, J.C., Kraft, N.A., Hale, D.: Outcomes of a community workshop to identify and rank barriers to the systematic literature review process. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE). ACM Press (2014)
9. Hernandes, E., Zamboni, A., Fabbri, S., Thommazo, A.D.: Using GQM and TAM to evaluate StArt – a tool that supports systematic review. CLEI Electronic Journal 15(1) (2012)
10. Kitchenham, B.A.: Guidelines for performing systematic literature reviews in software engineering. Tech. rep., Keele University (2007)
11. Kitchenham, B.A., Brereton, P.: A systematic review of systematic review process research in software engineering. Information and Software Technology 55(12), 2049–2075 (dec 2013)
12. Marchezan, L., Bolfe, G., Rodrigues, E., Bernardino, M., Basso, F.P.: Thoth: A web-based tool to support systematic reviews. In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE (2019)
13. Marshall, C., Brereton, P.: Systematic review toolbox. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE). ACM Press (2015)
14. Marshall, C., Brereton, P., Kitchenham, B.A.: Tools to support systematic reviews in software engineering a feature analysis. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE). ACM Press (2014)
15. Microsoft: Overview of the number of computer science publications (2021), `https://academic.microsoft.com/topic/41008148`
16. Molléri, J.S., Benitti, F.B.V.: SESRA. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE). ACM Press (2015)

17. Ramampiaro, H., Cruzes, D., Conradi, R., Mendona, M.: Supporting evidence-based software engineering with collaborative information retrieval. In: Proceedings of the 6th International Conference on Collaborative Computing (ICST). IEEE (2010)
18. Riaz, M., Sulayman, M., Salleh, N., Mendes, E.: Experiences conducting systematic reviews from novices' perspective. In: Proceedings of the 14th international conference on Evaluation and Assessment in Software Engineering (EASE). BCS Learning & Development (2010)
19. Ruiz-Rube, I., Person, T., Mota, J.M., Dodero, J.M., González-Toro, Á.R.: Evidence-based systematic literature reviews in the cloud. In: Proceedings of the 21st International Conference on Intelligent Data Engineering and Automated Learning (IDEAL). Springer International Publishing (2018)
20. Snyder, H.: Literature review as a research methodology: An overview and guidelines. Journal of Business Research (JBR) 104, 333–339 (2019)
21. Universitätsbibliothek Technische Universität München: Softwarevergleich Literaturverwaltung - 8. Aktualisierung (Juni 2020) (2021), https://mediatum.ub.tum.de/doc/1316333/1316333.pdf

All links were last followed on January 25, 2021.

Index