Jan Mendling
Haralambos Mouratidis (Eds.)

LNBIP 317

Information Systems in the Big Data Era

CAiSE Forum 2018
Tallinn, Estonia, June 11–15, 2018
Proceedings

Springer

Lecture Notes in Business Information Processing 317

More information about this series at http://www.springer.com/series/7911

Jan Mendling · Haralambos Mouratidis (Eds.)

Information Systems in the Big Data Era

CAiSE Forum 2018
Tallinn, Estonia, June 11–15, 2018
Proceedings

Springer

Editors
Jan Mendling
Wirtschaftsuniversität Wien
Vienna
Austria

Haralambos Mouratidis
University of Brighton
Brighton
UK

ISSN 1865-1348 ISSN 1865-1356 (electronic)
Lecture Notes in Business Information Processing
ISBN 978-3-319-92900-2 ISBN 978-3-319-92901-9 (eBook)
https://doi.org/10.1007/978-3-319-92901-9

Library of Congress Control Number: 2018944409

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at CAISE Forum 2018 held from June 28 to July 1, 2018, in Tallinn. CAISE is a well-established highly visible conference series on information systems engineering. The CAiSE Forum is a place within the CAiSE conference for presenting and discussing new ideas and tools related to information systems engineering. Intended to serve as an interactive platform, the forum aims at the presentation of emerging new topics and controversial positions, as well as demonstration of innovative systems, tools, and applications. The forum sessions at the CAiSE conference will facilitate the interaction, discussion, and exchange of ideas among presenters and participants. Contributions to the CAiSE 2018 Forum were welcome to address any of the conference topics and in particular the theme of this year's conference: "Information Systems in the Big Data Era." We invited two types of submissions:

- Visionary papers presenting innovative research projects, which are still at a relatively early stage and do not necessarily include a full-scale validation. Visionary papers are presented as posters in the forum.
- Demo papers describing innovative tools and prototypes that implement the results of research efforts. The tools and prototypes are presented as demos in the forum.

The management of paper submission and reviews was supported by the EasyChair conference system. There were 29 submissions, with 13 of them being nominated by the Program Committee (PC) chairs of the CAISE main conference. Each submission was reviewed by three PC members. The committee decided to accept 22 papers.

As chairs of the CAISE Forum, we would like to express again our gratitude to the PC for their efforts in providing very thorough evaluations of the submitted forum papers. We also thank the local organization team for their great support. Finally, we wish to thank all authors who submitted papers to CAISE and to the CAISE Forum.

April 2018

Jan Mendling
Haralambos Mouratidis

Organization

General Chairs

Marlon Dumas	University of Tartu, Estonia
Andreas Opdahl	University of Bergen, Norway

Organization Chair

Fabrizio Maggi	University of Tartu, Estonia

Program Committee Chairs

Jan Mendling	Wirtschaftsuniversität Wien, Austria
Haralambos Mouratidis	University of Brighton, UK

Program Committee

Raian Ali	Bournemouth University, UK
Said Assar	Institut Mines-Telecom, France
Saimir Bala	WU Vienna, Austria
Jan Claes	Ghent University, Belgium
Dirk Fahland	Eindhoven University of Technology, The Netherlands
Luciano García-Bañuelos	University of Tartu, Estonia
Haruhiko Kaiya	Shinshu University, Japan
Christos Kalloniatis	University of the Aegean, Greece
Dimka Karastoyanova	University of Groningen, The Netherlands
Henrik Leopold	Vrije Universiteit Amsterdam, The Netherlands
Daniel Lübke	Leibniz Universität Hannover, Germany
Massimo Mecella	Sapienza University of Rome, Italy
Selmin Nurcan	Université Paris 1 Panthéon, Sorbonne, France
Cesare Pautasso	University of Lugano, Switzerland
Michalis Pavlidis	University of Brighton, UK
Luise Pufahl	Hasso Plattner Institute, University of Potsdam, Germany
David Rosado	University of Castilla-La Mancha, Spain
Sigrid Schefer-Wenzl	FH Campus Vienna, Austria
Stefan Schönig	University of Bayreuth, Germany
Arik Senderovich	Technion, Israel
Arnon Sturm	Ben-Gurion University, Israel
Lucinéia Heloisa Thom	Federal University of Rio Grande do Sul, Brazil
Matthias Weidlich	Humboldt-Universität zu Berlin, Germany
Moe Wynn	Queensland University of Technology, Australia

Contents

Enabling Process Variants and Versions in Distributed Object-Aware Process Management Systems

Kevin Andrews(✉), Sebastian Steinau, and Manfred Reichert

Institute of Databases and Information Systems,
Ulm University, Ulm, Germany
{kevin.andrews,sebastian.steinau,manfred.reichert}@uni-ulm.de

Abstract. Business process variants are common in many enterprises and properly managing them is indispensable. Some process management suites already offer features to tackle the challenges of creating and updating multiple variants of a process. As opposed to the widespread activity-centric process modeling paradigm, however, there is little to no support for process variants in other process support paradigms, such as the recently proposed artifact-centric or object-aware process support paradigm. This paper presents concepts for supporting process variants in the object-aware process management paradigm. We offer insights into the distributed object-aware process management framework PHILharmonicFlows as well as the concepts it provides for implementing variants and versioning support based on *log propagation* and *log replay*. Finally, we examine the challenges that arise from the support of process variants and show how we solved these, thereby enabling future research into related fundamental aspects to further raise the maturity level of data-centric process support paradigms.

Keywords: Business processes · Process variants
Object-aware processes

1 Introduction

Business process models are a popular method for companies to document their processes and the collaboration of the involved humans and IT resources. However, through globalization and the shift towards offering a growing number of products in a large number of countries, many companies are face a sharp increase of complexity in their business processes [4,5,11]. For example, automotive manufacturers that, years ago, only had to ensure that they had stable processes for building a few car models, now have to adhere to many regulations for different countries, the increasing customization wishes of customers, and far faster development and time-to-market cycles. With the addition of Industry 4.0 demands, such as process automation and data-driven manufacturing, it is

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 1–15, 2018.
https://doi.org/10.1007/978-3-319-92901-9_1

becoming more important for companies to establish maintainable business processes that can be updated and rolled out across the entire enterprise as fast as possible.

However, the increase of possible process variants poses a challenge, as each additional constraint derived from regulations or product specifics either leads to larger process models or more process models showing different variants of otherwise identical processes. Both scenarios are not ideal, which is why there has been research over the past years into creating more maintainable process variants [5,7,11,13]. As previous research on process variant support has focused on activity-centric processes, our contribution provides a novel approach supporting *process variants in object-aware processes*. Similar to case handling or artifact-centric processes, object-aware processes are inherently more flexible than activity-centric ones, as they are less strictly structured, allowing for more freedom during process execution [1,3,6,10,12]. This allows object-aware processes to support processes that are very dynamic by nature and challenging to formulate in a sequence of activities in a traditional process model.

In addition to the conceptual challenges process variants pose in a centralized process server scenario, we examine how our approach contributes to managing the challenges of modeling and executing process variants on an architecture that can support scenarios with high scalability requirements. Finally, we explain how our approach can be used to enable updatable versioned process models, which will be essential for supporting schema evolution and ad-hoc changes in object-aware processes.

To help understand the notions presented in the contribution we provide the fundamentals of object-aware process management and process variants in Sect. 2. Section 3 examines the requirements identified for process variants. In Sect. 4 we present the concept for variants in object-aware processes as the main contribution of this paper. In Sect. 5 we evaluate whether our approach meets the identified requirements and discuss threats to validity as well as persisting challenges. Section 6 discusses related work, whereas Sect. 7 provides a summary and outlook on our plans to provide support for migrating running process instances to newer process model versions in object-aware processes.

2 Fundamentals

2.1 Object-Aware Process Management

PHILharmonicFlows, the object-aware process management framework we are using as a test-bed for the concepts presented in this paper, has been under development for many years at Ulm University [2,8,9,16,17]. This section gives an overview on the PHILharmonicFlows concepts necessary to understand the remainder of the paper. PHILharmonicFlows takes the basic idea of a data-driven and data-centric process management system and improves it by introducing the concept of *objects*. One such object exists for each business object present in a real-world business process. As can be seen in Fig. 1, a PHILharmonicFlows

object consists of data, in the form of *attributes*, and a state-based process model describing the *object lifecycle*.

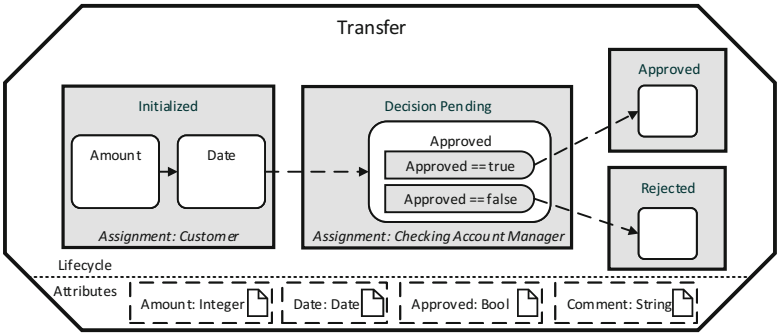


Fig. 1. Example object including lifecycle process

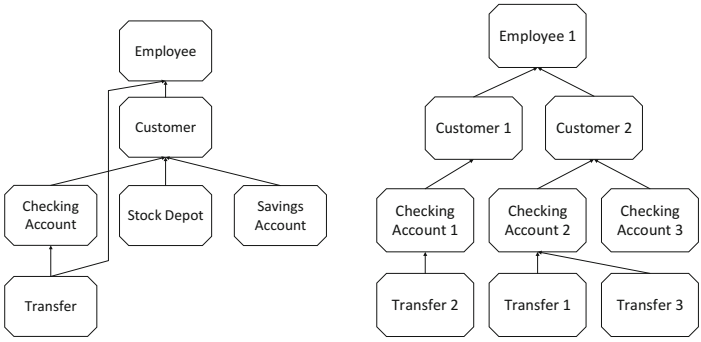
The attributes of the *Transfer* object (cf. Fig. 1) include *Amount*, *Date*, *Approval*, and *Comment*. The*lifecycle process*, in turn, describes the different *states (Initialized*, *Decision Pending*, *Approved*, and *Rejected)*, an instance of a *Transfer* object may have during process execution. Each state contains one or more *steps*, each referencing exactly one of the object attributes, thereby forcing that attribute to be written at run-time. The steps are connected by *transitions*, allowing them to be arranged in a sequence. The state of the object changes when all steps in a state are completed. Finally, alternative paths are supported in the form of *decision steps*, an example of which is the *Approved* decision step.

As PHILharmonicFlows is *data-driven*, the lifecycle process for the *Transfer* object can be understood as follows: The initial state of a *Transfer* object is *Initialized*. Once a *Customer* has entered data for the *Amount* and *Date* attributes, the state changes to *Decision Pending*, which allows an *Account Manager* to input data for *Approved*. Based on the value for *Approved*, the state of the *Transfer* object changes to *Approved* or *Rejected*. Obviously, this fine-grained approach to modeling a business process increases complexity when compared to the activity-centric paradigm, where the minimum granularity of a user action is one atomic activity or task, instead of an individual data attribute.

However, as an advantage, the object-aware approach allows for *automated form generation* at run-time. This is facilitated by the lifecycle process of an object, which dictates the attributes to be filled out before the object may switch to the next state, resulting in a personalized and dynamically created form. An example of such a form, derived from the lifecycle process in Fig. 1, is shown in Fig. 2.

Fig. 2. Example form

Note that a single object and its resulting form only constitutes one part of a complete PHILharmonicFlows process. To allow for complex executable business processes, many different objects and users may have to be involved [17]. It is noteworthy that *users* are simply special objects in the object-aware process management concept. The entire set of objects (including those representing users) present in a PHILharmonicFlows process is denoted as the *data model*, an example of which can be seen in Fig. 3a. At run-time, each of the objects can be instantiated to so-called *object instances*, of which each represents a concrete instance of an object. The lifecycle processes present in the various object instances are executable concurrently at run-time, thereby improving performance. Figure 3b shows a simplified example of an *instantiated data model* at run-time.



(a) Design-time (b) Run-time

Fig. 3. Data model

In addition to the objects, the data model contains information about the *relations* existing between them. A relation constitutes a logical association between two objects, e.g., a relation between a *Transfer* and a *Checking Account*. Such a relation can be instantiated at run-time between two concrete object instances of a *Transfer* and a *Checking Account*, thereby associating the two object instances with each other. The resulting meta information, i.e., the information that the *Transfer* in question belongs to a certain *Checking Account*, can be used to coordinate the processing of the two objects with each other.

Finally, complex object coordination, which becomes necessary as most processes consist of numerous interacting business objects, is possible in PHILharmonicFlows as well [17]. As objects publicly advertise their state information, the current state of an object can be utilized as an abstraction to coordinate with other objects corresponding to the same business process through a set of constraints, defined in a separate *coordination process*. As an example, consider a constraint stating that a *Transfer* may only change its state to *Approved* if there are less than 4 other *Transfers* already in the *Approved* state for one specific *Checking Account*.

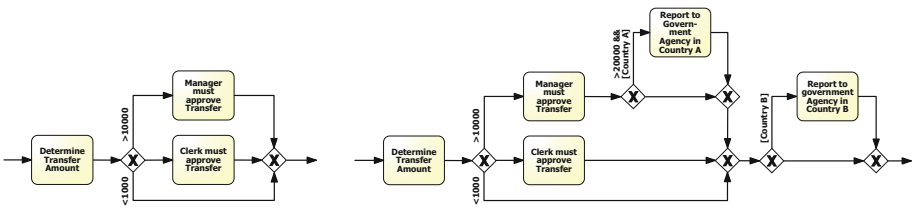
The various components of PHILharmonicFlows, i.e., objects, relations, and coordination processes, are implemented as microservices, turning PHILharmonicFlows into a fully distributed process management system for object-aware processes. For each object instance, relation instance, or coordination process instance one microservice is present at run-time. Each microservice only holds data representing the attributes of its object. Furthermore, the microservice only executes the lifecycle process of the object it is assigned to. The only information visible outside the individual microservices is the current "state" of the object, which, in turn, is used by the microservice representing the coordination process to properly coordinate the objects' interactions with each other.

2.2 Process Variants

Simply speaking, a process variant is one specific path through the activities of a process model, i.e., if there are three distinct paths to completing a business goal, three process variants exist. As an example, take the process of transferring money from one bank account to another, for which there might be three alternate execution paths. For instance, if the amount to be transferred is greater than \$10,000, a manager must approve the transfer, if the amount is less than \$10,000, a mere clerk may approve said transfer. Finally, if the amount is less than \$1,000, no one needs to approve the transfer. This simple decision on who has to approve the transfer implicitly creates three variants of the process.

As previously stated, modeling such variants is mostly done by incorporating them into one process model as alternate paths via choices (cf. Fig. 4a). As demonstrated in the bank transfer example, this is often the only viable option, because the amount to be transferred is not known when the process starts. Clearly, for more complex processes, each additional choice increases the complexity of the process model, making it harder to maintain and update.

To demonstrate this, we extend our previous example of a bank transfer with the addition of country-specific legal requirements for money transfers between accounts. Assuming the bank operates in three countries, A, B, and C, country A imposes the additional legal requirement of having to report transfers over \$20,000 to a government agency. On the other hand, Country B could require the reporting of all transfers to a government agency, while country C has no



(a) Base (b) Including extra Requirements

Fig. 4. Bank transfer process

such requirements. The resulting process model would now have to reflect all these additional constraints, making it substantially larger (cf. Fig. 4b).

Obviously, this new process model contains more information than necessary for its execution in one specific country. Luckily, if the information necessary to choose the correct process variant is available before starting the execution of the process, a different approach can be chosen: defining the various process variants as separate process models and choosing the right variant before starting the process execution. In our example this can be done as the country is known before the transfer process is started. Therefore, it is possible to create three country-specific process model variants, for countries A, B, and C, respectively. Consequently, each *process model variant* would only contain the additional constraints for that country not present in the *base process model*.

This reduces the complexity of the process model from the perspective of each country, but introduces the problem of having three different models to maintain and update. Specifically, changes that must be made to those parts of the model common to all variants, in our example the decision on who must approve the transfer, cause redundant work as there are now multiple process models that need updating. Minimizing these additional time-consuming workloads, while enabling clean variant-specific process models, is a challenge that many researchers and process management suites aim to solve [5,7,11,14,15].

3 Requirements

The requirements for supporting process variants in object-aware processes are derived from the requirements for supporting process variants in activity-centric processes, identified in our previous case studies and a literature review [5,7,11].

Requirement 1 *(Maintainability)***.** Enabling maintainability of process variants is paramount to variant management. Without advanced techniques, such as propagating changes made to a base process to its variants, optimizing a process would require changes in all individual variants, which is error-prone and time-consuming. To enable the features that improve maintainability, the base process and its variants must be structured as such (cf. Req. 2). Furthermore, process modelers must be informed if changes they apply to a base process introduce errors in the variants derived from them (cf. Req. 3).

Requirement 2 *(Hierarchical structuring)***.** As stated in Req. 1, a hierarchical structure becomes necessary between variants. Ideally, to further reduce workloads when optimizing and updating processes, the process variants of both lifecycle and coordination processes can be decomposed into further sub-variants. This allows those parts of the process that are shared among variants, but which are not part of the base process, to be maintained in an intermediate model.

Requirement 3 *(Error resolution)***.** As there could be countless variants, the system should report errors to process modelers automatically, as manual checking of all variants could be time-consuming. Additionally, to ease error resolution,

the concept should allow for the generation of resolution suggestions. To be able to detect which variants would be adversely affected by a change to a base model, automatically verifiable correctness criteria are needed, leading to Req. 4.

Requirement 4 *(Correctness).* The correctness of a process model must be verifiable at both design- and run-time. This includes checking correctness before a pending change is applied in order to judge its effects. Additionally, the effects of a change on process model variants must be determinable to support Req. 5.

Requirement 5 *(Scalability).* Finally, most companies that need process variant management solutions maintain many process variants and often act globally. Therefore, the solutions for the above requirements should be scalable, both in terms of computational complexity as well as in terms of the manpower necessary to apply them to a large number of variants. Additionally, as the PHILharmonicFlows architecture is fully distributed, we have to ensure that the developed algorithms work correctly in a distributed computing environment.

4 Variants and Versioning of Process Models

This section introduces our concepts for creating and managing different deployed versions and variants of data models as well as contained objects in an object-aware process management system. We start with the *deployment concept*, as the variant concept relies on many of the core notions presented here.

4.1 Versioning and Deployment Using Logs

Versioning of process models is a trivial requirement for any process management system. Specifically, one must be able to separate the model currently being edited by process modelers from the one used to instantiate new process instances. This ensures that new process instances can always be spawned from a stable version of the model that no one is currently working on. This process is referred to as *deployment*. In the current PHILharmonicFlows implementation, deployment is achieved by copying an *editable data model*, thereby creating a *deployed data model*. The deployed data model, in turn, can then be instantiated and executed while process modelers keep updating the editable data model.

As it is necessary to ensure that already running process instances always have a corresponding deployed model, the deployed models have to be *versioned* upon deployment. This means that the deployment operation for an editable data model labeled "M" automatically creates a deployed data model M_{T38} (*Data Model M, Timestamp 38*). Timestamp $T38$ denotes the logical timestamp of the version to be deployed, derived from the amount of modeling actions that have been applied in total. At a later point, when the process modelers have updated the editable data model M and they deploy the new version, the deployment operation gets the logical timestamp for the deployment, i.e., $T42$, and creates the deployed data model M_{T42} (*Data Model M, Timestamp* 42). As M_{T38} and M_{T42} are copies of the editable model M at the moment (i.e., timestamp) of

deployment, they can be instantiated and executed concurrently at run-time. In particular, process instances already created from M_{T38} should not be in conflict with newer instances created from M_{T42}.

The editable data model M, the two deployed models M_{T38} and M_{T42} as well as some instantiated models can be viewed in Fig. 5. The representation of each model in Fig. 5 contains the set of objects present in the model. For example, $\{X, Y\}$ denotes a model containing the two objects X and Y. Furthermore, the editable data model has a list of all modeling actions applied to it. For example, *L13:[+X]* represents the 13th modeling action, which added an object labeled "X". The modeling actions we use as examples throughout this section allow adding and removing entire objects. However, the concepts can be applied to any of the many different operations supported in PHILharmonicFlows, e.g., adding attributes or changing the coordination process.

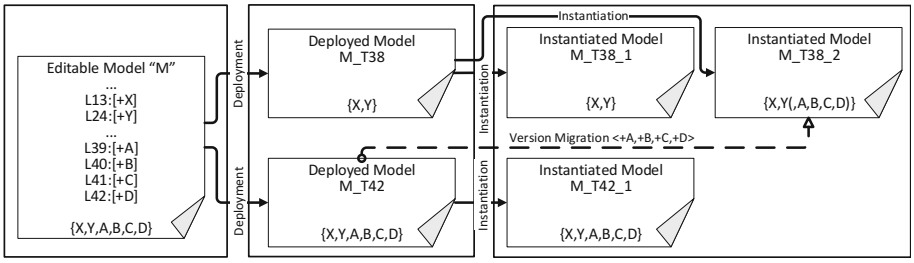


Fig. 5. Deployment example

To reiterate, versioned deployment is a basic requirement for any process management system and constitutes a feature that most systems offer. However, we wanted to develop a concept that would, as a topic for future research, allow for the migration of already running processes to newer versions. Additionally, as we identified the need for process variants (cf. Sect. 1), we decided to tackle all three issues, i.e., *versioned deployment*, *variants*, and *version migration of running processes*, in one approach.

Deploying a data model by simply cloning it and incrementing its version number is not sufficient for enabling version migration. Version migration requires knowledge about the changes that need to be applied to instances running on a lower version to migrate it to the newest version, denoted as $M_{T38} \Delta M_{T42}$ in our example. In order to obtain this information elegantly, we log all actions a process modeler completes when creating the editable model until the first deployment. We denote these log entries belonging to M as $logs\,(M)$. To create the deployed model, we replay the individual log entries $l \in logs\,(M)$ to a new, empty, data model. As all modeling actions are deterministic, this recreates the data model M step by step, thereby creating the deployed copy, which we denote as M_{T38}. Additionally, as replaying the logs in $logs\,(M)$ causes each modeling action to be repeated, the deployment process causes the deployed data

model M_{T38} to create its own set of logs, $logs\,(M_{T38})$. Finally, as data model M remains editable after a deployment, additional log entries may be created and added to $logs\,(M)$. Each consecutive deployment causes the creation of another deployed data model and set of logs, e.g. M_{T42} and $logs\,(M_{T42})$.

As the already deployed version, M_{T38} has its own set of logs, i.e., $logs\,(M_{T38})$, it is trivial to determine $M_{T38}\Delta M_{T42}$, as it is simply the *set difference*, i.e., $logs\,(M_{T42}) \setminus logs\,(M_{T38})$. As previously stated, $M_{T38}\Delta M_{T42}$ can be used later on to enable version migration, as it describes the necessary changes to instances of M_{T38} when migrating them to M_{T42}. An example of how we envision this concept functioning is given in Fig. 5 for the migration of the instantiated model M_{T38_2} to the deployed model M_{T42}.

To enable this logging-based copying and deployment of a data model in a distributed computing environment, the log entries have to be fitted with additional meta information. As an example, consider the simple log entry l_{42} which was created after a user had added a new object type to the editable data model:

$$l_{42} = \begin{cases} dataModelId & 6123823241189 \\ action & AddObjectType \\ params & [name : \text{“}Object\,D\text{”}] \\ timestamp & 42 \end{cases}$$

Clearly, the log entry contains all information necessary for its replay: the id of the data model or object the logged action was applied to, the type of action that was logged, and the parameters of this action. However, due to the distributed microservice architecture PHILharmonicFlows is built upon, a logical timestamp for each log entry is required as well. This timestamp must be unique and sortable across all microservices that represent parts of one editable data model, i.e., all objects, relations, and coordination processes. This allows PHILharmonicFlows to gather the log entries from the individual microservices, order them in exactly the original sequence, and replay them to newly created microservices, thereby creating a deployed copy of the editable data model.

Coincidentally, it must be noted that the example log entry l_{42} is the one created before deployment of M_{T42}. By labeling the deployment based on the timestamp of the last log entry, determining the modeling actions that need to be applied to an instance of M_{T38} to update it to M_{T42} can be immediately identified as the sequence $\langle l_{39}, l_{40}, l_{41}, l_{42}\rangle \subset logs\,(M_{T42})$, as evidenced by the example in Fig. 5.

4.2 Variants

As previously stated, we propose reusing the logging concept presented in Sect. 4.1 for creating and updating variants of data models and contained objects. In Sect. 4.1, we introduced two example data models, M_{T38} and M_{T42}, which were deployed at different points in time from the same editable data model. Additionally, we showed that the differences between these two deployed models are the actions applied by four log entries, namely $\langle l_{39}, l_{40}, l_{41}, l_{42}\rangle$. Expanding

upon this idea, we developed a concept for creating variants of data models using log entries for each modeling action, which we present in this section.

An example of our concept, in which two variants, $V1$ and $V2$, are created from the editable data model M, is shown in Fig. 6. The *editable base model*, M, has a sequence of modeling actions that were applied to it and logged in $\langle l_1, \ldots, l_{42} \rangle$. Furthermore, the two variants of M where created at different points in time, i.e., at different logical timestamps. Variant $V1$ was created at timestamp $T39$, i.e., the last action applied before creating the variant had been logged in l_{39}.

As we reuse the deployment concept for variants, the actual creation of a *data model variant* is, at first, merely the creation of an identical copy of the editable data model in question. For variant $V1$, this means creating an empty editable data model and replaying the actions logged in the log entries $\langle l_1, \ldots, l_{39} \rangle \subseteq logs\,(M)$, ending with the creation of object A. As replaying the logs to the new editable data model M_{V1} creates another set of logs, $logs\,(M_{V1})$, any further modeling actions that process modelers only apply to M_{V1} can be logged in $logs\,(M_{V1})$ instead of $logs\,(M)$. This allows us to add or remove elements not altered in the base model or other variants. An example is given by the removal of object A in $l_{40} \in logs\,(M_{V1})$, an action not present in $logs\,(M)$ or $logs\,(M_{V2})$.

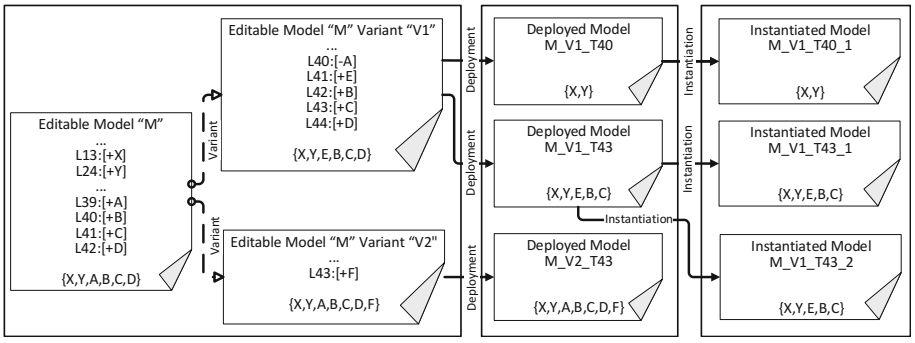


Fig. 6. Variant example

Up until this point, a variant is nothing more than a copy that can be edited independently of the base model. However, in order to provide a solution for maintaining and updating process variants (cf. Req. 1), the concept must also support the automated propagation of changes made to the base model to each variant. To this end, we introduce a hierarchical relationship between editable models, as required by Req. 2, denoted by $\sqsubset$. In the example (cf. Fig. 6), both variants are beneath data model M in the variant hierarchy, i.e., $M_{V1} \sqsubset M$ and $M_{V2} \sqsubset M$. For possible sub-variants, such as $M_{V2_{V1}}$, the hierarchical relationship is *transitive*, i.e., $M_{V2_{V1}} \sqsubset M \iff M_{V2} \sqsubset M$.

To fulfill Req. 1 when modeling a variant, e.g. $M_{V1} \sqsubset M$, we utilize the hierarchical relationship to ensure that all modeling actions applied to M are

propagated to M_{V1}, always ensuring that $logs\,(M_{V1}) \subseteq logs\,(M)$ holds. This is done by replaying new log entries added to $logs\,(M)$ to M_{V1}, which, in turn, creates new log entries in $logs\,(M_{V1})$. As an example, Fig. 7 shows the replaying of one such log, $l_{40} \in logs(M)$ to M_{V1}, which creates log entry $l_{42} \in logs\,(M_{V1})$.

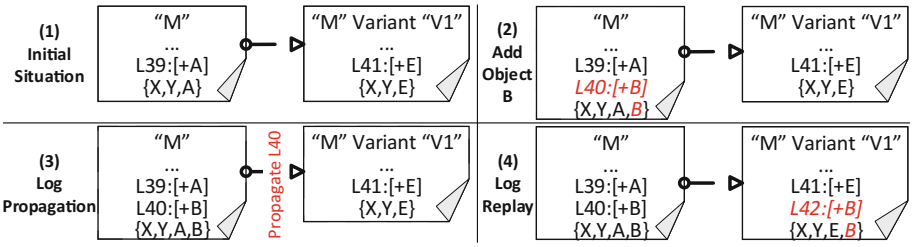


Fig. 7. Log propagation example

In the implementation, we realized this by making the propagation of the log entry for a specific modeling action part of the modeling action itself, thereby ensuring that updating the base model, including all variants, is atomic. However, it must be noted that, while the action being logged in both editable data models is the same, the logs have different timestamps. This is due to the fact that M_{V1} has the variant-specific log entries $\langle l_{40}, l_{41} \rangle \subset logs\,(M_{V1})$ and $l_{40} \in logs(M)$ is appended to the end of $logs\,(M_{V1})$ as $l_{42} \in logs\,(M_{V1})$. As evidenced by Fig. 6, variants created this way are fully compatible with the existing deployment and instantiation concept. In particular, from the viewpoint of the deployment concept, a variant is simply a normal editable model with its own set of logs that can be copied and replayed to a deployed model.

5 Evaluation

The presented concept covers all requirements (cf. Sect. 3) as we will show in the following. The main requirement, and goal of this research, was to develop a concept for maintainable process variants of object-aware data models and contained objects (cf. Req. 1). We managed to solve this challenge by introducing a strict hierarchical structure between the base data model, variants, and even sub-variants (cf. Req. 2). Furthermore, our solution ensures that the variants are always updated by changes made to their base models. As presented in Sect. 4.2, this is done by managing logs with logical timestamps and replaying them to variants that are lower in the hierarchy. This ensures that any modeling action applied to a variant will always take into consideration the current base model. However, this strict propagation of all modeling actions to all variants poses additional challenges. For instance, expanding on the situation presented in Fig. 6, a modeling action that changes part of the lifecycle process of object A could be logged as $l_{43} \in logs(M)$, causing the log to be replayed to variant

$V1$. However, $V1$ does not have an object A anymore, as is evidenced by the set of objects present, i.e., $\{X, Y, E, B, C, D\}$. Clearly, this is due to the fact that $l_{40} \in logs\,(M_{V1})$ removed object A from that variant.

As it is intentional for variant $V1$ to not comprise object A, this particular case poses no further challenge, as changes to an object not existing in a variant can be ignored by that variant. However, there are other scenarios to be considered, one of which is the application of modeling actions in a base model that have already been applied to a variant, such as introducing a transition between two steps in the lifecycle process of an object. If this transition already exists in a variant, the log replay to that variant will create an identical transition. As two transitions between the same steps are prohibited, this action would break the lifecycle process model of the variant and, in consequence, the entire object and data model it belongs to. A simplified example of the bank transfer object can be seen next to a variant with an additional transition between *Amount* and *Date* to showcase this issue in Fig. 8. The problem arises when trying to add a transition between *Amount* and *Date* to the base lifecycle process model, as the corresponding log entry gets propagated to the variant, causing a clash.

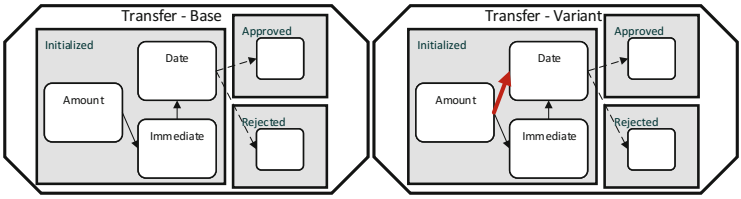


Fig. 8. Conflicting actions example

To address this and similar issues, which pose a *threat to validity* for our concept, we utilize the existing data model verification algorithm we implemented in the PHILharmonicFlows engine [16]. In particular, we leverage our distributed, micro-service based architecture to create clones of the parts of a variant that will be affected by a log entry awaiting application. In the example from Fig. 8, we can create a clone of the microservice serving the object, apply the log describing the transition between *Amount* and *Date,* and run our verification algorithm on the clone. This would detect any problem caused in a variant by a modeling action and generate an error message with resolution options, such as deleting the preexisting transition in the variant (cf. Reqs. 3 and 4). In case there is no problem with the action, we apply it to the microservice of the original object. How the user interface handles the error message (e.g., offering users a decision on how to fix the problem) is out of the scope of this paper, but has been implemented and tested as a proof-of-concept for some of the possible errors.

All other concepts presented in this paper have been implemented and tested in the PHILharmonicFlows prototype. We have headless test cases simulating a multitude of users completing randomized modeling actions in parallel, as well as

around 50,000 lines of unit testing code, covering various aspects of the engine, including the model verification, which, as we just demonstrated, is central to ensuring that all model variants are correct. Furthermore, the basic mechanism used to support variants, i.e., the creation of data model copies using log entries, has been an integral part of the engine for over a year. As we rely heavily on it for deploying and instantiating versioned data models (cf. Sect. 4.1), it is utilized in every test case and, therefore, thoroughly tested.

Finally, through the use of the microservice-based architecture, we can ensure that time-consuming operations, such as verifying models for compatibility with actions caused by log propagation, are highly scalable and cannot cause bottlenecks [2]. This would hardly be an issue at design-time either way, but we are ensuring that this basis for our future research into run-time version migration, or even migration between variants, is highly scalable (cf. Req. 5). Furthermore, the preliminary benchmark results for the distributed PHILharmonicFlows engine, running on a cluster of 8 servers with 64 CPUs total, are promising. As copying data models using logs is central to the concepts presented in this paper, we benchmarked the procedure for various data model sizes (5, 7, and 14 objects) and quadrupling increments of concurrently created copies of each data model. The results in Table 1 show very good scalability for the creation of copies, as creating 64 copies only takes twice as long as creating one copy. The varying performance between models of only slightly different size can be attributes to the fact that some of the more complex modeling operations are not yet optimized.

Table 1. Results

Example process	Objects	1 copy	4 copies	16 copies	64 copies
Recruitment	5	880 ms	900 ms	1120 ms	2290 ms
Intralogistics	7	2680 ms	2830 ms	4010 ms	9750 ms
Insurance	14	4180 ms	4470 ms	7260 ms	12170 ms

6 Related Work

Related work deals with modeling, updating, and managing of process variants in the activity-centric process modeling paradigm [5,7,11,13,15], as well as the management of large amounts of process versions [4].

The Provop approach [5] allows for flexible process configuration of large process variant collections. The activity-centric variants are derived from base processes by applying change operations. Only the set of change operations constituting the delta to the base process is saved for each variant, reducing the amount of redundant information. Provop further includes variant selection techniques that allow the correct variant of a process to be instantiated at run-time, based on the context the process is running in.

An approach allowing for the configuration of process models using questionnaires is presented in [13]. It builds upon concepts presented in [15], namely the

introduction of variation points in process models and modeling languages (e.g. C-EPC). A process model can be altered at these variation points before being instantiated, based on values gathered by the questionnaire. This capability has been integrated into the APROMORE toolset [14].

An approach enabling flexible business processes based on the combination of process models and business rules is presented in [7]. It allows generating ad-hoc process variants at run-time by ensuring that the variants adhere to the business rules, while taking the actual case data into consideration as well.

Focusing on the actual procedure of modeling process variants, [11] offers a decomposition-based modeling method for entire families of process variants. The procedure manages the trade-off between modeling multiple variants of a business process in one model and modeling them separately.

A versioning model for business processes that supports advanced capabilities is presented in [4]. The process model is decomposed into block fragments and persisted in a tree data structure, which allows versioned updates and branching on parts of the tree, utilizing the tree structure to determine affected parts of the process model. Unaffected parts of the tree can be shared across branches.

Our literature review has shown that there is interest in process variants and developing concepts for managing their complexity. However, existing research focuses on the activity-centric process management paradigm, making the current lack of process variant support in other paradigms, such as artifact- or data-centric, even more evident. With the presented research we close this gap.

7 Summary and Outlook

This paper focuses on the design-time aspects of managing data model variants in a distributed object-aware process management system. Firstly, we presented a mechanism for copying editable design-time data models to deployed run-time data models. This feature, by itself, could have been conceptualized and implemented in a number of different ways, but we strove to find a solution that meets the requirements for managing process variants as well. Secondly, we expanded upon the concepts created for versioned deployment to allow creating, updating, and maintaining data model variants. Finally, we showed how the concepts can be combined with our existing model verification tools to support additional requirements, such as error messages for affected variants.

There are still open issues, some of which have been solved for activity-centric process models, but likely require entirely new solutions for non-activity-centric processes. Specifically, one capability we intend to realize for object-aware processes is the ability to take the context in which a process will run into account when selecting a variant.

When developing the presented concepts, we kept future research into truly flexible process execution in mind. Specifically, we are currently in the process of implementing a prototypical extension to the current PHILharmonicFlows engine that will allow us to upgrade instantiated data models to newer versions. This kind of version migration will allow us to fully support schema evolution.

Additionally, we are expanding the error prevention techniques presented in our evaluation to allow for the verification of data model correctness for already instantiated data models at run-time. We plan to utilize this feature to enable ad-hoc changes of instantiated objects and data models, such as adding an attribute to one individual object instance without changing the deployed data model.

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

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. Andrews, K., Steinau, S., Reichert, M.: Towards hyperscale process management. In: Proceedings of the EMISA, pp. 148–152 (2017)
3. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. IEEE TCDE **32**(3), 3–9 (2009)
4. Ekanayake, C.C., La Rosa, M., ter Hofstede, A.H.M., Fauvet, M.-C.: Fragment-based version management for repositories of business process models. In: Meersman, R., et al. (eds.) OTM 2011. LNCS, vol. 7044, pp. 20–37. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25109-2_3
5. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. JSEP **22**(6–7), 519–546 (2010)
6. Hull, R.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Proceedings of the WS-FM, pp. 1–24 (2010)
7. Kumar, A., Yao, W.: Design and management of flexible process variants using templates and rules. Comput. Ind. **63**(2), 112–130 (2012)
8. Künzle, V.: Object-aware process management. Ph.D. thesis, Ulm University (2013)
9. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. JSME **23**(4), 205–244 (2011)
10. Marin, M., Hull, R., Vaculín, R.: Data centric BPM and the emerging case management standard: a short survey. In: Proceedings of the BPM, pp. 24–30 (2012)
11. Milani, F., Dumas, M., Ahmed, N., Matulevičius, R.: Modelling families of business process variants: a decomposition driven method. Inf. Syst. **56**, 55–72 (2016)
12. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30409-5
13. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: Configurable multiperspective business process models. Inf. Syst. **36**(2), 313–340 (2011)
14. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. **38**(6), 7029–7040 (2011)
15. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1), 1–23 (2007)
16. Steinau, S., Andrews, K., Reichert, M.: A modeling tool for PHILharmonicFlows objects and lifecycle processes. In: Proceedings of the BPMD (2017)
17. Steinau, S., Künzle, V., Andrews, K., Reichert, M.: Coordinating business processes using semantic relationships. In: Proceedings of the CBI, pp. 143–152 (2017)

Achieving Service Accountability Through Blockchain and Digital Identity

Fabrizio Angiulli, Fabio Fassetti, Angelo Furfaro(✉), Antonio Piccolo, and Domenico Saccà

DIMES - University of Calabria, P. Bucci, 41C, 87036 Rende, CS, Italy
{f.angiulli,f.fassetti,a.furfaro,a.piccolo}@dimes.unical.it, sacca@unical.it

Abstract. This paper proposes a platform for achieving accountability across distributed business processes involving heterogeneous entities that need to establish various types of agreements in a standard way. The devised solution integrates blockchain and digital identity technologies in order to exploit the guarantees about the authenticity of the involved entities' identities, coming from authoritative providers (e.g. public), and the trustiness ensured by the decentralized consensus and reliability of blockchain transactions.

Keywords: Service accountability · Blockchain · Digital identity

1 Introduction

In last few years, the number of contracts, transactions and other forms of agreements among entities has grown mainly thanks to the pervasiveness of ICT technologies which eased and speed up the business interactions. However, such growth has not been followed up by suitable technological innovations for that regards important issues like the need for accountability in agreements. Thus, the first problem to tackle is that of handling services where many actors, possibly belonging to independent organizations and different domains, need to base their interactions on "strong" guarantees of reliability and not on mutual trust or on reputation systems.

We, then, aim at defining an innovative platform for handling cooperative processes and services where the assumption of responsibility and the attribution of responsibility concerning activities performed by the involved actors can be clearly and certifiably stated. The platform should assure *trust* and *accountability* to be applied at different steps of service supply, from message exchange to transaction registration, till automatic execution of contract clauses.

Technologies for centralized handling of services are mature and widely employed, conversely open problems arise when the management is distributed or decentralized and there is the need to guarantee reliability and security of services.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 16–23, 2018.
https://doi.org/10.1007/978-3-319-92901-9_2

First of all, there is the *consensus* problem. Although exploiting a trusted and certified third-part for certifying identities is reasonable and acceptable by all the involved parts, the details about all the events concerning processes and services handled by the platform cannot be tackled by assuming the presence of a central trusted coordinator which would be one of the involved parts due to the intrinsic nature of decentralization and to need for a strong trust level guaranteed by distributed consensus. Many research efforts have been devoted to this issue and the state-of-the-art decentralized cooperation model is the blockchain.

Blockchain technology was early developed for supporting bitcoin cryptocurrency. Such technology allows the realization of a *distributed ledger* which guarantees a distributed consensus and consists in an asset database shared across a network of multiple sites, geographies or institutions. All participants within a network can have their own identical copy of the ledger [1]. The technology is based on a P2P approach, the community collaborates to obtain an agreed and reliable version of the ledger, where all the transactions are signed by authors and publicly visible, verified and validated. The actors of the transactions are identified by a public key representing their blockchain address, thus, there is no link between transaction actor in the ledger and his real-world identity. One of the main contribution of the proposed platform is the providing of a suitable solution to overcome this limitation.

The second main problem to tackle is the *accountability* in cooperative services. The mechanism of identity/service provider based on the SAML 2 protocol [2] represents a valid solution for handling digital identities through a standard, authoritative, certified, trusted, public entity. Towards this direction, the European Community introduced the *eIDAS* regulation [3] and the member States developed their own identity provider system accordingly (for example the Italian Public System for Digital Identity (SPID) [4]). However, how to embed the accountability in cooperative services in order to state responsibility and to certify activities of involved subjects is still a challenging problem. Solving this issue is a fundamental step for achieving a trustable and accountable infrastructure. Note that since the blockchain can be public readable, this could potentially arise a privacy problem that should be taken suitably into account. The main contribution of the work is, then, the definition of a platform aimed at handling services and processes involving different organizations of different domains that guarantees (*i*) *privacy*, (*ii*) *accountability*, (*iii*) *no third-part trustiness*.

The rest of the paper is organized as follows. Section 2 presents the preliminary notions about blockchain and digital identities technologies. Section 3 illustrates the peculiarities of the considered scenario and the related issues. Section 4 presents the detail about the proposed platform. Finally, Sect. 5 draws the conclusions.

2 Preliminary Notions

Bitcoin [5] is a digital currency in which encryption techniques are used to verify the transfer of funds, between two users, without relying on a central bank.

Transactions are linked each other through a hash of characters in one block, that references a hash in another block. Blocks chained and linked together are saved in a distributed database called blockchain. Changes made in one location get propagated throughout the blockchain ledger for anyone to verify that there is no double spending. The process of verification, Proof of Work (PoW), is carried out by some members of the network called miners using the power of specialized hardware to verify the transactions and to create a new block every 10 min. The miner is compensated in cryptocurrency that can be exchanged for fiat money, products, and services.

The success of Bitcoin encouraged the spawning of a group of alternative currencies, or "altcoins", using the same general approach but with different optimizations and tweaks. A breakthrough was introduced at the beginning of 2015 when Blockchain 2.0 comes in introducing new features, among which the capability to run decentralized applications inside the blockchain. In most cases, protection against the problem of double spending is still ensured by a Proof of Work algorithm. Some projects, instead, introduced a more energy efficient approach called Proof of Stake (PoS). In particular, PoS is a kind of algorithm by which a cryptocurrency blockchain network aims to achieve distributed consensus. In PoS based blockchains the creator of the next block is chosen in a deterministic way, and the chance that an account is chosen depends on its wealth, for example the quantity of stake held. The forging of a new block can be rewarded with the creation of new coins or with transaction fees only [6]. Some of the new terminology introduced by the Blockchain 2.0 involves the terms: Smart Contracts or DAPPs (decentralized applications), Smart Property and DAOs (decentralized autonomous organizations). Typically a contract involves two parties, and each party must trust the other party to fulfill its side of the obligation. Smart contracts remove the need for one type of trust between parties because its behaviour is defined and automatically executed by the code. In fact, a smart contract is defined as being autonomous, self-sufficient and decentralized [7]. The general concept of smart property is to control the ownership and the access of an asset by having it registered as a digital asset on the blockchain, identified by an address, the public key, and managed by its private key. Property could be physical assets (home, car, or computer), or intangible assets (reservations, copyrights, etc.). When a DAPP adopts more complicated functionalities such as public governance on the blockchain and mechanisms for financing its operations, like crowdfundings, it turns into a DAO (decentralized autonomous organization) [8,9].

In short, Blockchain 1.0 is limited to currency for digital payment systems, while Blockchain 2.0 is also being used for critical applications like contracts used for market, economic and financial applications. The most successful Blockchain 2.0 project is represented by Ethereum, the, so called, world computer [10].

2.1 Public Digital Identity Provider

The public identity provider mechanism is based on the Security Assertion Markup Language (SAML) protocol [2] which is an open-standard defined by

the OASIS Security Services Technical Committee. The latest version is the 2.0 released in 2005 which allows web-based authentication and authorization implementing the single sign-on (SSO) access control policy.

The main goal of the protocol is exchanging authentication and authorization data between parties.

The SAML protocol introduces three main roles: (*i*) the client (said *principal*) who is the entity whose identity has to be assessed to allow the access to a given resource/service; (*ii*) the *identity provider* (IDP) who is in charge of identifying the client asking for a resource/service, stating that such client is known to the IDP and providing some information (attributes) about the client; (*iii*) the *service provider* (SP) who is the entity in charge of providing a resource/service to a client after a successful authentication phase through a interaction with an IDP who provide client attributes to the SP. Thus, the resource/service access control flow can be summarized as follows:

1. the client requires for a resource/service to the service provider;
2. the service provider requests an IDP for an identity assertion about the requiring client;
3. the service provider makes the access control decision basing on the received assertion.

3 Scenario and Issues

In order to illustrate the advantages of the devised platform and to discuss the adopted solutions, firstly we present peculiarities and issues in the scenarios where the platform could constitute a valid and useful framework.

As previously introduced, the platform is particularly suited when the handled underlying process (for example a business process) involves different entities/companies/organizations that cooperate and want to work together without trusting on each other.

We assume to deal with two main entities: (*i*) one or more companies that cooperate in a common business involving a distributed and heterogeneous process where the accountability of the transactions of a primary importance (e.g. complex supply chains, logistics of hazardous substances); (*ii*) users and supervisors working in the corporates which are equipped with a public digital identity (denoted as *pub-ID* in the following) and a blockchain address (denoted as *bc-ID* in the following). We want to accomplish the following desiderata:

1. having the guaranty that a given transaction T happened from an entity X and an entity Y;
2. having the guaranty about the real-world entities X and Y that have performed T;
3. importantly, the above guarantees should be provided without trusting on any intermediary or third-part authoritative entity.

This corresponds to achieve the following objectives:

Goal 1. *Privacy:* each non-authorized entity should not know any detail about happened transactions.
Goal 2. *Accountability:* each authorized entity should know the real-world entity behind an actor performing a transaction.
Goal 3. *No third-part trustiness:* each entity should not need to trust on a component owned by another entity involved in the business process.

With these goals in mind, the proposed platform exploits the peculiarities of two technologies: *BlockChain* (BC) and *Public Digital Identity Providers*. As for the latter technology the adopted solution exploits the IDP/SP mechanism based on the OASIS SAML standard for exchanging authentication and authorization [2]. The blockchain ensures the trustiness about the effective execution of the transactions stored in the distributed ledger and allows us the accomplishment of goals 1 and 3. The IDP/SP mechanism allows us to obtain the real-world entity behind an account without the need of trusting on authentication mechanisms related to companies. Thus, this allows us to accomplish goal 2.

Since blockchain is like a distributed and shared database and, then, each node in the network can read the contents of the blocks and since a business process may contain sensitive data, the platform should allow to exploits the advantages of blockchains with no harm to the privacy of the data.

The IDP provides some information about the real-world entities associated with the account. However, more specific information are often needed in order to manage the privileges of reading/writing data. This can easily be taken into account thanks to the attribute provider mechanisms which is natively integrated in the IDP/SP mechanisms through the definition of Attribute Providers owned by the companies.

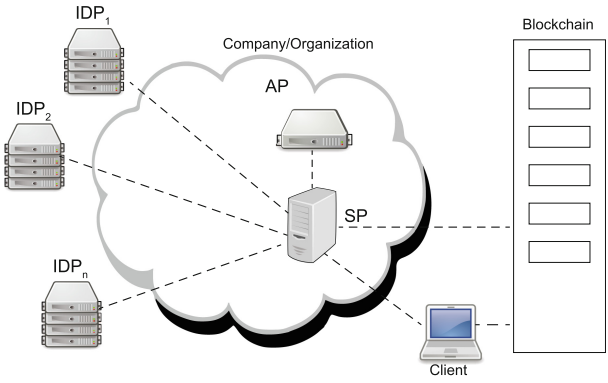


Fig. 1. The proposed architecture

4 Platform

The proposed architecture devoted at accomplishing the goals depicted in the previous section is reported in Fig. 1. The basic processes of the platform are described in details in the following sections. The main actors of the platform are:

- *Public Identity Provider (IDP).* The platform assumes the presence of one or more public IDPs which constitute an *external* authoritative source of information about the digital identity of the entities involved in the business process handled by the platform and, then, are controlled neither by such entities nor by the service provider but are trusted by all.
- *Service Provider (SP).* A relevant role in the platform is played by the service provider which constitutes an *internal* resource and it is in charge of handling business process details and sensitive data about involved entities; thus, it is expected that each company/organization builds its own SP for managing its data and it is not required that a company/organization trusts on the SP of another company/organization.
- *Attribute Provider (AP).* In the platform also one or more attribute providers can be present. Such services provide additional information about the entities accessing the platform through their public digital identity for example concerning roles and privileges with respect to business process data.
- *Blockchain (BC).* One of the fundamental component of the platform is the blockchain 2.0, which is external to the platform, supports smart contracts enactment and execution and implements the distributed ledger.
- *Client.* Each company member involved in the business process represents a client that has to register in the platform through its public digital identity and interacts with the blockchain through its blockchain address.

4.1 User Registration

The first step is to register the company users in the platform. Each company can independently register its members to the platform by defining its own Service Provider. It is required that the SP writes on the blockchain by creating a *smart contract* stating the association between the public digital identity of the member and its BC address. In turn, the member has to invoke this smart contract to confirm such pairing.

Note that it is not required that other companies trust on the SP producing the smart contract, since it can always be checked if the association is true. Indeed, the pairing between pub-ID and bc-ID can be checked by any SP by requiring the user to access the SP through its pub-ID and then to prove it is the owner of the bc-ID by signing a given challenge. The task of *member verification* performed by the service provider will be detailed in Sect. 4.3.

4.2 Smart Contract Handling

This section illustrates the platform core service consisting in exploiting the block-chain to record both business process transactions and the involved actors.

This process is started by the SP that creates a smart contract SC which records the hash of the business process transaction T and the bc-IDs of those actors which are (possibly with different roles) involved in T. To make the transaction effective, each actor has to confirm its participation by invoking an ad-hoc method of SC to confirm his agreement to play the role assigned to him by the SP. This accomplishes the three main goals planned for the platform.

Privacy Issue. Recording the hash of T allows us to ensure the suitable privacy level about the subject of the transaction. Indeed, in this way, the SP which has created the SC owns the transaction data. So, in order for an entity E to know the details of the accomplished transaction, it has to authenticate itself at the SP through its public digital identity and to require the data. The SP can, thus, verify the identity of E and check its privileges w.r.t. T. Optionally, the SP could also record the access request on the blockchain by asking E to invoke a suitable method on an ad-hoc smart contract.

Accountability Issue. From the smart contract SC, the entity E can get the hash of the transaction T and the bc-IDs of the involved actors. The entity E can also get from the blockchain the hash of the pairing about the bc-IDs of these actors and their pub-IDs. The pairing associated with this hash is stored by the SP, which can provide E this information if and only if E has privileges on T.

No Need for Trusted Third-Part Authority Issue. As for this issue, since both the hash of the data of transaction T and the hash of the pairings between bc-IDs and pub-IDs of the involved actors are recorded on the blockchain, each entity E can always check if all the information coming from the SP are valid, without needing of trusting on it. Note that, if E is one of the actors involved in T, E must be able to perform this check before invoking the method on the smart contract associated with T required for confirming T.

4.3 Service Provider Tasks

In this section, we describe the main features that a service provider should offer to be suitably embedded in the platform. We assume that the service provider is registered on one or more public IDPs which handle the public digital identities of the actors involved in the business process that should be managed by the platform. The service provider accomplishes three main tasks: *members registration*, *members verification* and *privileges management*.

Member Registration. The service provider allows a member to register by communicating with the IDPs according to the registration process described in Sect. 4.1. Once getting the bc-ID of the member, the service provider produces a smart contract containing the association between bc-ID and pub-ID which has to be confirmed by the user invoking a suitable method on the smart contract.

Member Verification. Through a service provider, an entity can check the pairing between bc-ID and pub-ID of a member by performing the following step. First, the service provider asks the member to access through its pub-ID on a public

IDP. Second, the service provider asks the member to prove that he is the owner of the bc-ID by requiring him to encode a challenge string (for example a human-readable sentence) with the private key associated with the bc-ID.

Privileges Management. The service provider which owns pairings and transactions can provide the details about them just to entities authorized at knowing such details. This can be accomplished by requiring that (*i*) the entity authenticates itself through its pub-ID on a public IDP and (*ii*) that the privileges of the entity returned by the attribute provider are enough to allow the access.

5 Conclusions

The orchestration of cooperative services is becoming the standard way to implement innovative service provisions and applications emerging in many contexts like e-government and e-procurement. In this scenario, technological solutions have been developed which address typical issues concerning cooperation procedures and data exchange. However, embedding accountability inside distributed and decentralized cooperation models is still a challenging issue. In this work, we devise a suitable approach to guarantee the service accountability which is based on state-of-art solutions regarding digital identity and distributed consensus technologies for building a distributed ledger. In particular, the proposal exploits the notion of smart contracts as supported by blockchain 2.0.

This work has been partially supported by the "IDService" project (CUP B28117000120008) funded by the Ministry of Economic Development under Grant Horizon 2020 - PON I&C 2014-20 and by the project P.O.R. "SPID Advanced Security - SPIDASEC" (CUP J88C17000340006).

References

1. Hancock, M., Vaizey, E.: Distributed ledger technology: beyond block chain. Technical report GS/16/1, UK Government Chief Scientific Adviser (2016)
2. Cantor, S., Kemp, J., Maler, E., Philpott, R.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.02. OASIS Standard Specification (2005)
3. Bender, J.: eIDAS regulation: EID - opportunities and risks (2015)
4. AgID - Agenzia per lItalia Digitale: Spid - regole tecniche (2017)
5. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. White paper (2008)
6. Popov, S.: A probabilistic analysis of the Nxt forging algorithm. Ledger **1**, 69–83 (2016)
7. Tapscott, D., Tapscott, A.: Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. Portfolio, London (2016)
8. Buterin, V.: Bootstrapping a decentralized autonomous corporation: part I. Bitcoin Mag. (2013)
9. Buterin, V.: DAOs are not scary, part 1 and 2. Bitcoin Mag. (2014)
10. Buterin, V.: A next-generation smart contract and decentralized application platform. White paper (2014)

CrowdCorrect: A Curation Pipeline for Social Data Cleansing and Curation

Amin Beheshti[1,2(✉)], Kushal Vaghani[1], Boualem Benatallah[1], and Alireza Tabebordbar[1]

[1] University of New South Wales, Sydney, Australia
{sbeheshti,z5077732,boualem,alirezat}@cse.unsw.edu.au
[2] Macquarie University, Sydney, Australia
amin.beheshti@mq.edu.au

Abstract. Process and data are equally important for business process management. Data-driven approaches in process analytics aims to value decisions that can be backed up with verifiable private and open data. Over the last few years, data-driven analysis of how knowledge workers and customers interact in social contexts, often with data obtained from social networking services such as Twitter and Facebook, have become a vital asset for organizations. For example, governments started to extract knowledge and derive insights from vastly growing open data to improve their services. A key challenge in analyzing social data is to understand the raw data generated by social actors and prepare it for analytic tasks. In this context, it is important to transform the raw data into a contextualized data and knowledge. This task, known as data curation, involves identifying relevant data sources, extracting data and knowledge, cleansing, maintaining, merging, enriching and linking data and knowledge. In this paper we present *CrowdCorrect*, a data curation pipeline to enable analysts cleansing and curating social data and preparing it for reliable business data analytics. The first step offers automatic feature extraction, correction and enrichment. Next, we design micro-tasks and use the knowledge of the crowd to identify and correct information items that could not be corrected in the first step. Finally, we offer a domain-model mediated method to use the knowledge of domain experts to identify and correct items that could not be corrected in previous steps. We adopt a typical scenario for analyzing Urban Social Issues from Twitter as it relates to the Government Budget, to highlight how CrowdCorrect significantly improves the quality of extracted knowledge compared to the classical curation pipeline and in the absence of knowledge of the crowd and domain experts.

1 Introduction

Data analytics for insight discovery is a strategic priority for modern businesses [7,11]. Data-driven approaches in process analytics aims to value decisions that can be backed up with verifiable private and open data [10]. Over the last

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 24–38, 2018.
https://doi.org/10.1007/978-3-319-92901-9_3

few years, data-driven analysis of how knowledge workers and customers interact in social contexts, often with data obtained from social networking services such as Twitter (twitter.com/) and Facebook (facebook.com/), have become a vital asset for organizations [15]. In particular, social technologies have transformed businesses from a platform for private data content consumption to a place where social network workers actively contribute to content production and opinion making. For example, governments started to extract knowledge and derive insights from vastly growing open data to improve their services.

A key challenge in analyzing social data is to understand the raw data generated by social actors and prepare it for analytic tasks [6,12,14]. For example, tweets in Twitter are generally unstructured (contain text and images), sparse (offer limited number of characters), suffer from redundancy (same tweet retweeted) and prone to slang words and misspellings. In this context, it is important to transform the raw data (e.g. a tweet in Twitter or a Post in Facebook) into a contextualized data and knowledge. This task, known as data curation, involves identifying relevant data sources, extracting data and knowledge, cleansing (or cleaning), maintaining, merging, enriching and linking data and knowledge.

In this paper we present *CrowdCorrect*, a data curation pipeline to enable analysts cleansing and curating social data and preparing it for reliable data analytics. The first step offers automatic feature extraction (e.g. keywords and named entities), correction (e.g. correcting misspelling and abbreviation) and enrichment (e.g. leveraging knowledge sources and services to find synonyms and stems for an extracted/corrected keyword). In the second step, we design micro-tasks and use the knowledge of the crowd to identify and correct information items that could not be corrected in the first step. For example, social workers usually use abbreviations, acronyms and slangs that cannot be detected using automatic algorithms. Finally, in the third step, we offer a domain-model mediated method to use the knowledge of domain experts to identify and correct items that could not be corrected in previous steps. The contributions of this paper are respectively three-folds:

- We provides a customizable approach for extracting raw social data, using feature-based extraction. A feature is an attribute or value of interest in a social item (such as a tweet in Twitter) such as a keyword, topic, phrase, abbreviation, special characters (e.g. '#' in a tweet), slangs, informal language and spelling errors. We identify various categories for features and implement micro-services to automatically perform major data curation tasks.
- We design and implement micro-tasks to use the knowledge of the crowd and to identify and correct extracted features. We present an algorithm to compose the proposed micro-services and micro-tasks to curate the tweets in Twitter.
- We offer a domain-model mediated method to use the knowledge of domain experts to identify and correct items that could not be corrected in previous steps. This domain model presented as a set of rule-sets for a specific domain (e.g. Health) and will be used in cases where the automatic curation algorithms and the knowledge of the crowd were not able to properly contextualize the social items.

CrowdCorrect is offered as an open source project, that is publicly available on GitHub[1]. We adopt a typical scenario for analyzing Urban Social Issues from Twitter as it relates to the Australian government budget[2], to highlight how CrowdCorrect significantly improves the quality of extracted knowledge compared to the classical curation pipeline and in the absence of knowledge of the crowd and domain experts. The remainder of this paper is organized as follows. Section 2 represents the background and the related work. In Sect. 3 we present the overview and framework for the CrowdCorrect curation pipeline and present the three main data processing elements: Automatic Curation, Crowd Correction, and Domain Knowledge Reuse. In Sect. 4 we present the motivating scenario along with the experiment and the evaluation. Finally, we conclude the paper with a prospect on future work in Sect. 5.

2 Background and Related Work

The continuous improvement in connectivity, storage and data processing capabilities allow access to a data deluge from open and private data sources [2,9,39]. With the advent of widely available data capture and management technologies, coupled with social technologies, organizations are rapidly shifting to datafication of their processes. Social Network Analytics shows the potential and the power of computation to improve products and services in organizations. For example, over the last few years, governments started to extract knowledge and derive insights from vastly growing open data to improve government services, predict intelligence activities, as well as to improve national security and public health [37].

At the heart of Social Data Analytics lies the data curation process: This consists of tasks that transform raw social data (e.g. a tweet in Twitter which may contain text and media) into curated social data (contextualized data and knowledge that is maintained and made available for use by end-users and applications). Data curation involves identifying relevant data sources, extracting data and knowledge, cleansing, maintaining, merging, enriching and linking data and knowledge. The main step in social data curation would be to clean and correct the raw data. This is vital as for example in Twitter, with only 140 characters to convey your thoughts, social workers usually use abbreviations, acronyms and slangs that cannot be detected using automatic machine learning (ML) and Natural Language Processing (NLP) algorithms [3,13].

Social networks have been studied fairly extensively in the general context of analyzing interactions between people, and determining the important structural patterns in such interactions [3]. More specifically and focusing on Twitter [30], there have been a large number of work presenting mechanisms to capture, store, query and analyze Twitter data [23]. These works focus on understanding various aspects of Twitter data, including the temporal behavior of tweets arriving in a Twitter [33], measuring user influence in twitter [17], measuring message

[1] https://github.com/unsw-cse-soc/CrowdCorrect.
[2] http://www.budget.gov.au/.

propagation in Twitter [44], sentiment analysis of Twitter audiences [5], analyzing Twitter data using Big Data Tools and techniques [19], classification of tweets in twitter to improve information filtering [42] (including feature-based classification such as topic [31] and hashtag [22]), feature extraction from Twitter (include topic [45], and keyword [1], named entity [13] and Part of Speech [12] extraction).

Very few works have been considering cleansing and correcting tweets in Twitter. In particular, data curation involves identifying relevant data sources, extracting data and knowledge [38], cleansing [29], maintaining [36], merging [27], summarizing, enriching [43] and linking data and knowledge [40]. For example, information extracted from tweets (in Twitter) is often enriched with metadata on geo-location, in the absence of which the extracted information would be difficult to interpret and meaningfully utilize. Following, we briefly discuss some related work focus on curating Twitter data. Duh et al. [20] highlighted the need for curating the tweets but did not provide a framework or methodology to generate the contextualized version of a tweet. Brigadir et al. [16] presented a recommender system to support curating and monitoring lists of Twitter users. There has been also some annotated corpus proposed to normalize the tweets to understand the emotions [35] in a tweet, identify mentions of a drug in a tweet [21] or detecting political opinions in tweets [32]. The closest work in this category to our approach is the noisy-text[3] project, which does not provide the crowd and domain expert correction step.

Current approaches in Data Curation rely mostly on data processing and analysis algorithms including machine learning-based algorithms for information extraction, item classification, record-linkage, clustering, and sampling [18]. These algorithms are certainly the core components of data-curation platforms, where high-level curation tasks may require a non-trivial combination of several algorithms [4]. In our approach to social data curation, we specifically focus on cleansing and correcting the raw social data; and present a pipeline to apply curation algorithms (automatic curation) to the information items in social networks and then leverage the knowledge of the crowd as well as domain experts to clean and correct the raw social data.

3 CrowdCorrect: Overview and Framework

To understand the social data and supporting the decision making process, it is important to correct and transform raw social data generated on social networks into contextualized data and knowledge that is maintained and made available for use by analysts and applications. To achieve this goal, we present a data curation pipeline, CrowdCorrect, to enable analysts cleansing and curating social data and preparing it for reliable business data analytics. Figure 1 illustrates an overview of the CrowdCorrect curation pipeline, consist of three main data processing elements: Automatic Curation, Crowd Correction, and Domain Knowledge Reuse.

[3] https://noisy-text.github.io/norm-shared-task.html.

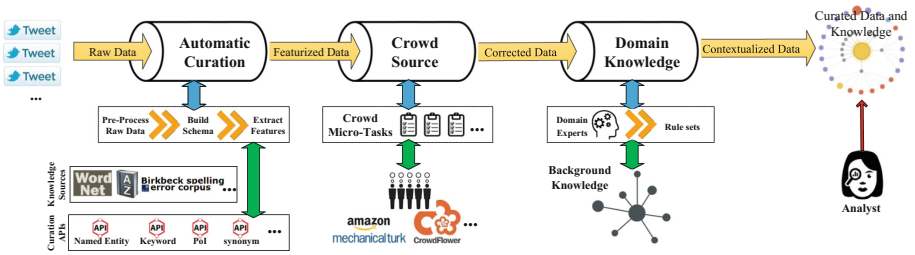


Fig. 1. Curation pipeline for cleansing and correcting social data.

3.1 Automatic Curation: Cleansing and Correction Tasks

Data cleansing or data cleaning deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data [34]. In the context of social networks, this task is more challenging as social workers usually use abbreviations, acronyms and slangs that cannot be detected using learning algorithms. Accordingly cleansing and correcting social raw data is of high importance. In the automatic curation (first step in the CrowdCorrect pipeline), we first develop services to ingest the data from social networks. At this step, we design and implement three services: to ingest and persist the data, to extract features (e.g. keywords) and to correct them (e.g. knowledge sources and services such as dictionaries and wordNet).

Ingestion Service. We implement ingestion micro-services (for Twitter, Facebook, GooglePlus and LinkedIn) and make them available as open source to obtain and import social data for immediate use and storage in a database. These services will automatically persist the data in CoreDB [8], a data lake service and our previous work. CoreDB enable us to deal with social data: this data is large scale, never ending, and ever changing, arriving in batches at irregular time intervals. We define a schema for the information items in social networks (such as Twitter, Facebook, GooglePlus and LinkedIn) and persist the items in MongoDB (a data island in our data lake) in JSON (json.org/) format, a simple easy to parse syntax for storing and exchanging data. For example, according to the Twitter schema, a tweet in Twitter may have attributes such as: (i) text: The text of a tweet; (ii) geo: The location from which a tweet was sent; (iii) hashtags: A list of hashtags mentioned in a tweet; (iv) domains: A list of the domains from links mentioned in a tweet; (v) lang: The language a tweet was written in, as identified by Twitter; (vi) links: A list of links mentioned in a tweet; (vii) media.type: The type of media included a tweet; (viii) mentions: A list of Twitter usernames mentioned in a tweet; and (ix) source: The source of the tweet. For example, 'Twitter for iPad'.

Extraction Services. We design and implement services to extract items from the content of unstructured items and attributes. To achieve this goal, we

propose data curation feature engineering: this refers to characterizing variables that grasp and encode information, thereby enabling to derive meaningful inferences from data. We propose that features will be implemented and available as uniformly accessible data curation Micro-Services: functions implementing features. These features include, but not limited to:

- Lexical features: words or vocabulary of a language such as Keyword, Topic, Phrase, Abbreviation, Special Characters (e.g. '#' in a tweet), Slangs, Informal Language and Spelling Errors.
- Natural-Language features: entities that can be extracted by the analysis and synthesis of Natural Language (NL) and speech; such as Part-Of-Speech (e.g. Verb, Noun, Adjective, Adverb, etc.), Named Entity Type (e.g. Person, Organization, Product, etc.), and Named Entity (i.e., an instance of an entity type such as 'Malcolm Turnbull'[4] as an instance of entity type Person).
- Time and Location features: the mentions of time and location in the content of the social media posts. For example in Twitter the text of a tweet may contain a time mention '3 May 2017' or a location mention 'Sydney; a city in Australia'.

Correction Services. We design and implement services to use the extracted features in previous step and to identify and correct the misspelling, jargons (i.e. special words or expressions used by a profession or group that are difficult for others to understand) and abbreviations. These services leverage knowledge sources and services such as WordNet (wordnet.princeton.edu/), STANDS4 (abbreviations.com/abbr_api.php) service to identify acronyms and abbreviations, Microsoft cognitive-services[5] to check the spelling and stems, and cortical (cortical.io/) service to identify jargons. The result of this step (automatic curation) will be an annotated dataset which contain the cleaned and corrected raw data. Figure 2 illustrates an example of an automatically curated tweet.

3.2 Manual Curation: Crowd and Domain-Experts

Social items, e.g. a tweet in Twitter, are commonly written in forms not conforming to the rules of grammar or accepted usage. Examples include abbreviations, repeated characters, and misspelled words. Accordingly, social items become text normalization challenges in terms of selecting the proper methods to detect and convert them into the most accurate English sentences [41]. There are several existing text cleansing techniques which are proposed to solve the issues, however they possess some limitations and still do not achieve good results overall. Accordingly, crowdsourcing [24] techniques can be used to obtain the knowledge of the crowd as an input into the curation task and to tune the automatic curation phase (previous step in the curation pipeline).

[4] https://en.wikipedia.org/wiki/Malcolm_Turnbull.
[5] https://azure.microsoft.com/en-au/try/cognitive-services/my-apis/.

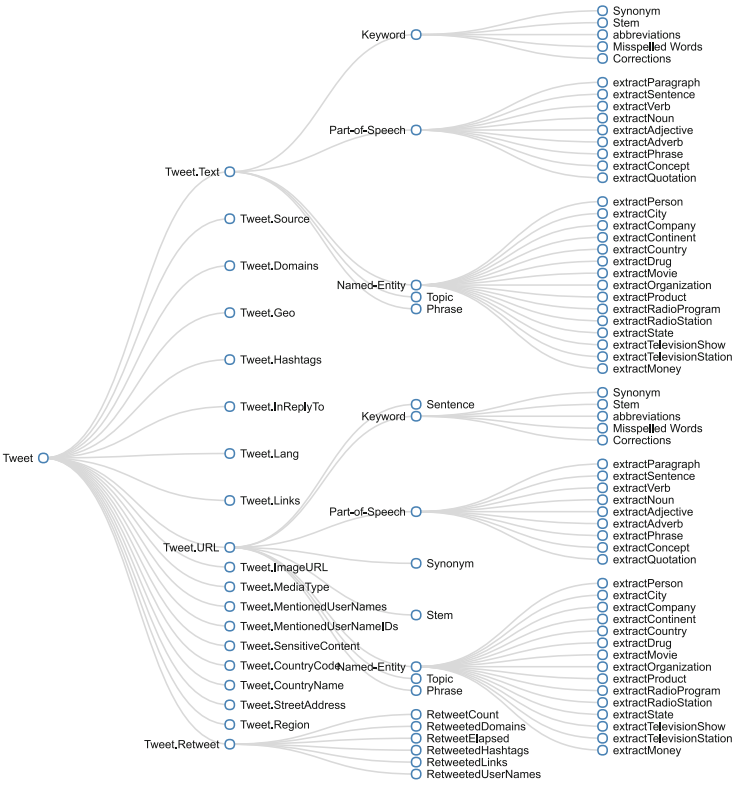


Fig. 2. An example of an automatically curated tweet.

Crowd Correction. Crowdsourcing rapidly mobilizes large numbers of people to accomplish tasks on a global scale [26]. For example, anyone with access to the Internet can perform micro-tasks [26] (small, modular tasks also known as Human Intelligence Tasks) on the order of seconds using platforms such as Amazon's Mechanical Turk (mturk.com), crowdflower (crowdflower.com/). It is also possible to use social services such as Twitter Polls[6] or simply designing a Web-based interface to share the micro-tasks with friends and colleagues. In this step, we design a simple Web-based interface to automatically generating the micro-tasks to share with people and use their knowledge to identify and correct information items that could not be corrected in the first step; or to verify if such automatic correction was valid. The goal is to have a hybrid combinations of crowd workers and automatic algorithmic techniques that may result in building collective intelligence We have designed two types of crowd micro-tasks [26]: suggestion and correction tasks.

Suggestion Micro-tasks. We design and implement an algorithm to present a tweet along with an extracted feature (e.g. a keyword extracted using the

[6] https://help.twitter.com/en/using-twitter/twitter-polls.

extraction services in Sect. 3.1) to the crowd and ask the crowd worker if the extracted feature can be considered as misspelled, abbreviation, or jargon. Algorithm 1 illustrates how we automatically generate suggestion micro-tasks.

```
Data: Automatically Curated Social-Item
Result: Annotated Social-Item
Extract Features from Social-Item;
array Question-Set ["misspelled ?", "abbreviations ?", "jargon ?"];
for Each feature in Extracted-Feature do
    for Each question in Question-Set do
        Generate Suggestion Micro-Task as follows:
        Display Social-Item;
        Display feature;
        Display question;
        Retrieve the Crowd Feedback (Yes/No);
        Annotate the feature (e.g. "misspelled" or "not misspelled");
    end
end
```

Algorithm 1. Automatically generating Suggestion Micro-Tasks.

Correction Micro-tasks. We design and implement an algorithm to present a tweet along with the features that has been identified as misspelled, abbreviation, or jargon in the previous crowd task; to the crowd and ask for the correct form of the feature. For example, if a keyword (in a tweet) identified as misspelled, we demonstrate the tweet, the (misspelled) keyword, a set of automatic corrections/suggestions (generated using correction services in Sect. 3.1) to the crowd. The crowd will be asked to select the correct suggestion or input a new correction if needed. Algorithm 2 illustrates how we automatically generate correction micro-tasks. Figure 3, in Sect. 4.1, illustrates examples of suggestion and correction micro-tasks.

3.3 Domain Knowledge Reuse

The goal of previous steps in the curation pipeline, is to identify the misspells, abbreviation and jargons and cook the social item (e.g. a tweet) to be usable in the analytics task. Although the automatic and crowd correction steps turn the raw data into a contextualized data, still there will be cases where the crowd are not able to correct the features. For example, there may be cases where the meaning of a keyword or an abbreviations is uncertain: there could be more than one meaning and the crowd or the automatic algorithm may not be able to provide the correct suggestion. For example, in the tweet "They gave me the option of AKA or limb salvage. I chose the latter.", the automatic and crowd correction tasks can identify AKA as an abbreviation, however providing correct replacement for this term requires the domain knowledge in health. A domain expert in health, i.e. the person who has the background knowledge

```
Data: Annotated Social-Item (Suggestion Micro-Tasks)
Result: Corrected Social-Item
Extract Features and Annotations from Annotated Social-Item;
for Each feature in Extracted-Feature do
    for Each annotation in Annotation-Set do
        if annotation = ("misspelled" OR "abbreviations" OR "jargon") then
            Generate Correction Micro-Task as follows:
            Display Social-Item;
            Display feature;
            Correction-Set = Correction-Service(feature);
            Display Correction-Set; Display Question("Choose/Input the
            correct" + annotation);
        else
            Annotate the Social-Item("No Need for Manually Correction");
        end
    end
end
```

Algorithm 2. Automatically generating Correction Micro-Tasks.

and experience in health, can identify people, organization, and items - such as diseases, drugs, devices, jobs and more; may be able to understand that (in this tweet) the AKA stands for 'Above-knee amputation'.

To address this challenge, we offer a domain-model mediated method to use the knowledge of domain experts to identify and correct items that could not be corrected in previous steps. To achieve this goal, and as an ongoing and future work, we are designing micro-tasks to illustrate an item (e.g. a tweet) to a domain expert (e.g. in health) and ask if the item is related to that specific domain or not. If the domain expert verify the item as related to the domain, then we use the PageRank[7] algorithm to measure the importance of features in that domain. Moreover, we will use A-Priori Algorithm [28] to eliminate most large feature sets as candidates by looking first at smaller feature sets and recognizing that a large set cannot be frequent unless all its subsets are.

This domain model will be presented as a set of rule-sets (i.e. association rule) for a specific domain (e.g. Health) and will be used in cases where the automatic curation algorithms and the knowledge of the crowd were not able to properly contextualize the social items. The form of an association rule is $I \rightarrow j$, where I is a set of features and j is a social item. The implication of this association rule is that if all of the social items in I appear in some domain (e.g. health), then j is 'likely' to appear in that domain as well. As an ongoing work we will define the confidence of the rule $I \rightarrow j$ to be the ratio of the support for $I \cup j$ to the support for I. We will design and implement algorithms to find association rules with high confidence.

[7] https://en.wikipedia.org/wiki/PageRank.

4 Motivating Scenario and Experiment

Social media networks create huge opportunities in helping businesses build relationships with customers, gain more insights into their customers, and deliver more value to them. Despite all the advantages of Twitter use, the content generated by Twitter users, i.e. tweets, may not be all that useful if they contain irrelevant and incomprehensible information, therefore making it difficult to analyse. To understand this challenge, we present a scenario in social network analytics and we consider the analytics task related to *"understanding a Governments' Budget in the context of Urban Social Issues"*. A typical governments' budget denote how policy objectives are reconciled and implemented in various categories and programs. In particular, budget categories (e.g. 'Health', 'Social-Services', 'transport' and 'employment') defines a hierarchical set of programs (e.g. 'Medicare Benefits' in Health, and 'Aged Care' in Social-Services). These programs refers to a set of activities or services that meet specific policy objectives of the government [25]. Using traditionally adopted budget systems, it would be difficult to accurately evaluate the governments' services requirements and performance. For example, it is paramount to stabilize the economy through timely and dynamic adjustment in expenditure plans by considering related *social issues*.

The first step in this task would be to decide if a tweet is related to a budget category (e.g. Health) or not. This task is challenging and requires extracting features such as keywords from a tweet, correct it using knowledge sources and services, and also to cover cases where algorithms cannot correctly identify and curate the features, we need to go one step further and leverage the knowledge of the crowd. After these steps, machine learning algorithms may classify the tweet as related to the specific budget category.

4.1 Experiment

The Australian Government budget sets out the economic and fiscal outlook for Australia, and shows the Government's social and political priorities. The Treasurer handed down the Budget 2016–17 at 7.30pm on Tuesday 3 May, 2016. To properly analyze the proposed budget, we have collected all tweets from one month before and two months after this date. In particular, for these three months, we have selected 15 million tweets, persisted and indexed in the MongoDB (mongodb.com/) database using the ingestion services (Sect. 3.1).

Then we use the extraction services (Sect. 3.1) to extract keywords from the text of the tweets. After this step, we use the correction services (Sect. 3.1) to identify the misspells, abbreviations, or jargons and replace them with the correct form to generate a new automatically curated version of the tweet. Then, we used the algorithms presented in Sect. 3.2, to access the tweets in the data lake and automatically construct the crowd micro-tasks. Figure 3, illustrates screenshots of our Web-based tool (crowdcorrect.azurewebsites.net/), which automatically generate the Suggestion Crowd Micro-Task and Correction Crowd Micro-Task.

Fig. 3. Screenshot of our Web-based tool, which automatically generate the suggestion crowd micro-task (A) and correction crowd micro-task (B).

In our experiment, we have asked students enrolled in semester two 2017 in the Web Application Engineering course[8] and some members of the Service Oriented Computing group in the University of New South Wales, Australia to participate in the crowd tasks. We also share the Web-based crowd tool with people on Twitter using hashtags such as '*#crowd*' and '*#crowdsourcing*'. Finally, we invited a set of crowd users from a local organization (www.westpac.com.au/): these users were co-workers which met the criteria. More than hundred participants contributed to the crowd-tasks.

To construct a domain mediated model, we construct a simple crowd micro-task, to present a tweet to a domain expert (in this experiment, the person who has the background knowledge and experience in health, can identify people, organization, and items - such as diseases, drugs, devices, jobs and more - related to health; and can verify if a tweet is related to health or not) and ask if the tweet is related to health. If the tweet has been considered related to health, the tweet will be added into the grand truth dataset and can be used as the training data for the machine learning algorithms, e.g. a classifier that may decide if a tweet is related to health or not.

4.2 Evaluation

We evaluated `CrowdCorrect` over Twitter data using *effectiveness*, achieving a high quality result in terms of precision and recall metric. The effectiveness is determined with the standard measures precision, recall and F-measure. Precision is the number of correctly identified tweets divided by the total number of tweets, recall is the number of correctly identified tweets divided by the total number of related tweets, and F-measure is the harmonic mean of precision and recall. Let us assume that TP is the number of true positives, FP the number of false positives (wrong results), TN the number of true negatives, and FN the number of false negatives (missing results). Then, Precision$= \frac{TP}{TP+FP}$, Recall$= \frac{TP}{TP+FN}$, and F-measure$= \frac{2*Precision*Recall}{Precision+Recall}$.

We have used three months of Twitter data as the input dataset from May 2016 to August 2016, which it's size was roughly around fifteen million tweets. To demonstrate the effectiveness of the proposed method, we created two datasets in the field of health care. The first dataset contains raw tweets, and the second

[8] https://webapps.cse.unsw.edu.au/webcms2/course/index.php?cid=2465.

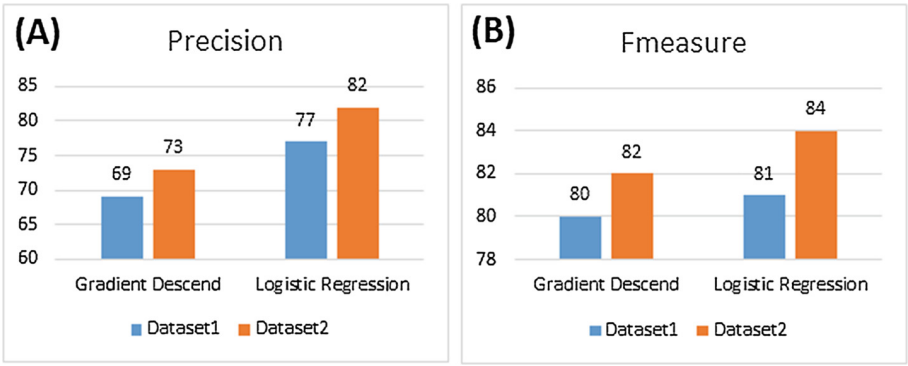


Fig. 4. Measure of improvement in recall (A) and F-measure (B).

dataset contains curated tweets (over 5000 automatically and manually curated tweets), where all Jargons, misspells and abbreviations are identified and corrected through the automatic and manual steps in the curtion pipeline.

To test the applicability of the proposed approach, we created four classifiers using a binomial logistic regression algorithm and a gradient descend algorithm. A logistic regression classifier is a generalized linear model that we can use to model or predict categorical outcome variables. As an example, we can consider a tweet related or not related to the health category of the budget. On the other side, gradient descend algorithm is widely used in optimization problems, and aims to minimize a cost function. This algorithm starts with a set of random input parameter and then it iteratively moves the parameter values to minimize the cost function.

Discussion. In this experiment, the classifiers have been constructed to verify if a tweet is relevant to 'health care' or not. First we trained two classifiers (Logistic Regression and gradient descend algorithms) using the raw and curated tweets. For training classifiers, we filtered out tokens occurred for less than three times. We also removed punctuation and stop words, and we used porter stemmer for stemming the remaining tokens. As illustrated in Fig. 4(A), both logistic regression and gradient descend algorithm outperformed in the curated dataset. In particular, the gradient descend algorithm has improved the precision by 4%, and the amount of improvement using the logistic regression algorithm is 5%. In addition, Fig. 4(B) illustrates the measure improvement in F-measure: the F-measure has improved in both gradient descend classifier and logistic regression classifier by 2% and 3% respectively.

5 Conclusion

Nowadays, an enormous amount of user-generated data is published continuously on a daily basis. Social media sites such as Twitter and Facebook have

empowered everyone to post and share information on just about any topic. Consequently, this data contains a rich; hidden pulse of information for analysts, governments and organizations. Understanding this data is therefore vital and a priority. However, this is not a trivial task. Let's take twitter as an example. Tweets are generally unstructured (e.g. text, images), sparse (e.g. tweets have only 140 characters), suffer from redundancy (e.g. same tweet re-tweeted) and prone to slang words and misspellings. As such, this raw data needs to be contextualized by a data curation pipeline before fed into for deeper analytics. In this paper, we proposed a curation pipeline to enable analysts cleansing and curating social data and preparing it for relatable data analytics. We investigated and proposed a hybrid combinations of crowd workers and automatic algorithmic techniques that may result in building collective intelligence. we presented a scenario in understanding a Governments' Budget in the context of Urban Social Issues. We evaluated our approach by measuring accuracy of classifying a tweet corpus before and after incorporating our approach.

Acknowledgements. We Acknowledge the Data to Decisions CRC (D2D CRC) and the Cooperative Research Centres Program for funding this research.

References

1. Abilhoa, W.D., De Castro, L.N.: A keyword extraction method from Twitter messages represented as graphs. Appl. Math. Comput. **240**, 308–325 (2014)
2. Abu-Salih, B., Wongthongtham, P., Beheshti, S., Zhu, D.: A preliminary approach to domain-based evaluation of users' trustworthiness in online social networks. In: 2015 IEEE International Congress on Big Data, New York City, NY, USA, 27 June–2 July 2015, pp. 460–466 (2015)
3. Aggarwal, C.C.: An introduction to social network data analytics. In: Social Network Data Analytics, pp. 1–15 (2011)
4. Anderson, M., et al.: Brainwash: a data system for feature engineering. In: CIDR (2013)
5. Bae, Y., Lee, H.: Sentiment analysis of Twitter audiences: measuring the positive or negative influence of popular Twitterers. J. Assoc. Inf. Sci. Technol. **63**(12), 2521–2535 (2012)
6. Batarfi, O., Shawi, R.E., Fayoumi, A.G., Nouri, R., Beheshti, S., Barnawi, A., Sakr, S.: Large scale graph processing systems: survey and an experimental evaluation. Cluster Comput. **18**(3), 1189–1213 (2015)
7. Beheshti, A., Benatallah, B., Motahari-Nezhad, H.R.: ProcessAtlas: a scalable and extensible platform for business process analytics. Softw. Pract. Exp. **48**(4), 842–866 (2018)
8. Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V.M., Xiong, H., Zhao, X.: Coredb: a data lake service. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, 06–10 November 2017, pp. 2451–2454 (2017)
9. Beheshti, S., Benatallah, B., Motahari-Nezhad, H.R.: Galaxy: a platform for explorative analysis of open data sources. In: Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, 15–16 March 2016, pp. 640–643 (2016)

10. Beheshti, S., Benatallah, B., Motahari-Nezhad, H.R.: Scalable graph-based OLAP analytics over process execution data. Distrib. Parallel Databases **34**(3), 379–423 (2016)
11. Beheshti, S.-M.-R., Benatallah, B., Sakr, S., Grigori, D., Motahari-Nezhad, H.R., Barukh, M.C., Gater, A., Ryu, S.H.: Process Analytics - Concepts and Techniques for Querying and Analyzing Process Data. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-25037-3
12. Beheshti, S., Benatallah, B., Venugopal, S., Ryu, S.H., Motahari-Nezhad, H.R., Wang, W.: A systematic review and comparative analysis of cross-document coreference resolution methods and tools. Computing **99**(4), 313–349 (2017)
13. Beheshti, S., Tabebordbar, A., Benatallah, B., Nouri, R.: On automating basic data curation tasks. In: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, 3–7 April 2017, pp. 165–169 (2017). https://doi.org/10.1145/3041021.3054726
14. Beheshti, S., Venugopal, S., Ryu, S.H., Benatallah, B., Wang, W.: Big data and cross-document coreference resolution: current state and future opportunities. CoRR abs/1311.3987 (2013)
15. Beheshti, S., et al.: Business process data analysis. In: Beheshti, S., et al. (eds.) Process Analytics, pp. 107–134. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-25037-3_5
16. Brigadir, I., Greene, D., Cunningham, P.: A system for Twitter user list curation. In: Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 293–294. ACM (2012)
17. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in Twitter: the million follower fallacy. ICWSM **10**(10–17), 30 (2010)
18. Chai, X., et al.: Social media analytics: the Kosmix story. IEEE Data Eng. Bull. **36**(3), 4–12 (2013)
19. Chitrakala, S.: Twitter data analysis. In: Modern Technologies for Big Data Classification and Clustering, p. 124 (2017)
20. Duh, K., Hirao, T., Kimura, A., Ishiguro, K., Iwata, T., Yeung, C.M.A.: Creating stories: social curation of Twitter messages. In: ICWSM (2012)
21. Ginn, R., Pimpalkhute, P., Nikfarjam, A., Patki, A., OConnor, K., Sarker, A., Smith, K., Gonzalez, G.: Mining Twitter for adverse drug reaction mentions: a corpus and classification benchmark. In: Proceedings of the Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing (2014)
22. Godin, F., Slavkovikj, V., De Neve, W., Schrauwen, B., Van de Walle, R.: Using topic models for Twitter hashtag recommendation. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 593–596. ACM (2013)
23. Goonetilleke, O., Sellis, T., Zhang, X., Sathe, S.: Twitter analytics: a big data management perspective. SIGKDD Explor. Newsl. **16**(1), 11–20 (2014). https://doi.org/10.1145/2674026.2674029
24. Howe, J.: The rise of crowdsourcing. Wired Mag. **14**(6), 1–4 (2006)
25. Kim, N.W., et al.: BudgetMap: engaging taxpayers in the issue-driven classification of a government budget. In: CSCW, pp. 1026–1037 (2016)
26. Kittur, A., Nickerson, J.V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., Horton, J.: The future of crowd work. In: CSCW (2013)
27. Kooge, E., et al.: Merging data streams. Res. World **2016**(56), 34–37 (2016)
28. Koyutürk, M., Grama, A., Szpankowski, W.: An efficient algorithm for detecting frequent subgraphs in biological networks. Bioinformatics **20**(Suppl_1), i200–i207 (2004)

29. Krishnan, S., et al.: Towards reliable interactive data cleaning: a user survey and recommendations. In: HILDA@ SIGMOD, p. 9 (2016)
30. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: WWW (2010)
31. Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A., Choudhary, A.: Twitter trending topic classification. In: 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW), pp. 251–258. IEEE (2011)
32. Maynard, D., Funk, A.: Automatic detection of political opinions in tweets. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) ESWC 2011. LNCS, vol. 7117, pp. 88–99. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25953-1_8
33. Perera, R.D., Anand, S., Subbalakshmi, K., Chandramouli, R.: Twitter analytics: architecture, tools and analysis. In: Military Communications Conference, 2010-MILCOM 2010, pp. 2186–2191. IEEE (2010)
34. Rahm, E., Do, H.H.: Data cleaning: problems and current approaches. IEEE Data Eng. Bull. **23**(4), 3–13 (2000)
35. Roberts, K., Roach, M.A., Johnson, J., Guthrie, J., Harabagiu, S.M.: EmpaTweet: annotating and detecting emotions on Twitter. In: LREC, vol. 12, pp. 3806–3813 (2012)
36. Rundensteiner, E., et al.: Maintaining data warehouses over changing information sources. Commun. ACM **43**(6), 57–62 (2000)
37. Russom, P., et al.: Big data analytics. TDWI Best Practices Report, Fourth Quarter, pp. 1–35 (2011)
38. Sadeghi, F., et al.: VisKE: visual knowledge extraction and question answering by visual verification of relation phrases. In: CVPR, pp. 1456–1464. IEEE (2015)
39. Salih, B.A., Wongthongtham, P., Beheshti, S.M.R., Zajabbari, B.: Towards a methodology for social business intelligence in the era of big social data incorporating trust and semantic analysis. In: Second International Conference on Advanced Data and Information Engineering (DaEng-2015). Springer, Bali (2015)
40. Shen, W., et al.: Entity linking with a knowledge base: issues, techniques, and solutions. ITKDE **27**(2), 443–460 (2015)
41. Sosamphan, P., et al.: SNET: a statistical normalisation method for Twitter. Master's thesis (2016)
42. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in Twitter to improve information filtering. In: SIGIR. ACM (2010)
43. Troncy, R.: Linking entities for enriching and structuring social media content. In: WWW, pp. 597–597 (2016)
44. Ye, S., Wu, S.F.: Measuring message propagation and social influence on Twitter.com. In: Bolc, L., Makowski, M., Wierzbicki, A. (eds.) SocInfo 2010. LNCS, vol. 6430, pp. 216–231. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16567-2_16
45. Zhao, W.X., Jiang, J., He, J., Song, Y., Achananuparp, P., Lim, E.P., Li, X.: Topical keyphrase extraction from Twitter. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 379–388. Association for Computational Linguistics (2011)

Service Discovery and Composition in Smart Cities

Nizar Ben-Sassi[1], Xuan-Thuy Dang[1], Johannes Fähndrich[1], Orhan-Can Görür[2], Christian Kuster[1], and Fikret Sivrikaya[1(✉)]

[1] GT-ARC gGmbH, Berlin, Germany
{nizar.ben-sassi,fikret.sivrikaya}@gt-arc.com
[2] DAI-Labor, TU Berlin, Berlin, Germany

Abstract. The ongoing digitalization trend has given rise to the concept of smart cities, targeting the interconnection of city infrastructure and services over a digital layer for innovative technological solutions as well as improvements on existing facilities in cities. This article investigates the critical information system constituents of smart cities that facilitate the holistic integration of its ecosystem and resources with the aim to foster dynamic and adaptive software. We identify three main enablers in this direction: (i) semantic functional description of city objects, representing physical devices or abstract services, (ii) a distributed service directory that embodies available city services for service lookup and discovery, (iii) planning tools for selecting and chaining basic services to compose new complex services. We provide an overview of the approach adopted in our ongoing smart city project for each of these three dimensions.

Keywords: Smart cities · Service discovery · Service composition
Semantic service description · Adaptive planning

1 Introduction

The perennial trend of urbanization has been transforming human life for many decades, whereby the city infrastructure and services become integral parts of our lives in the form of transportation systems, energy systems, and many more. More recently, the rising trend of digitalization brings new dimensions to urbanization; a concept typically captured by the term "smart cities". Advancements in information and communication technologies (ICT), such as the Internet of Things (IoT) and cloud computing provides a foundation for the digitalization of city systems [1]. This often results in better resource utilization among infrastructure providers and more flexible interaction between citizens, authorities and other stakeholders through ubiquitous access to information. However, the abundance of city data makes information management one of the central challenges in enabling cross-domain collaboration. With the lack of unified data and service integration platforms, the transformation to digital cities often results in specific applications that represent isolated service and data silos.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 39–48, 2018.
https://doi.org/10.1007/978-3-319-92901-9_4

In general, the information systems of a smart city can be in the form of physical devices with ICT capabilities, entirely virtual online services, or a combination of both, which we commonly refer to as *Smart City Objects (SCO)* in this article. A SCO can be as simple as a temperature sensor providing data to the cloud or as complex as a trip assistance service that stems from a well-crafted composition of many other SCOs from the transport, environment, and other city domains. The project "Intelligent Framework for Service Discovery and Composition" (ISCO)[1], presented in this paper, aims to develop an open and extensible platform together with supporting tools for the creation and holistic interconnection of SCOs from different sectors and stakeholders. This objective helps move away from fragmented IoT solutions and isolated data silos in cities towards a more integrated and harmonized smart city ecosystem. ISCO enables stakeholders from diverse domains to provide novel, efficient and dynamic services, optionally composed of other existing services in the smart city. One of the main challenges of such a platform is ensuring its scalability while enabling efficient development, deployment, and discovery of smart city objects.

In the remainder of this paper, we present a high-level overview of the ISCO solution (Sect. 2), followed by its main architectural components: a unified semantic model for SCOs based on semantic web technologies (Sect. 3), a scalable distributed architecture for SCO discovery based on information centric networking (Sect. 4), and a two-level planning approach for orchestrating through both generic and application-specific service ontologies (Sect. 5).

2 Background and ISCO Approach

The smart city concept has been attracting strong attention of the research community from a large variety of research fields, particularly in the computer science and information systems disciplines [2,3]. On the other hand, a growing number of smart city initiatives is spawned around the world in recent years [4]. Given the vast breadth and depth of the smart city scope, we do not intend to provide a comprehensive background here, and refer the readers to the surveys on the topic, e.g., [2,5], for a detailed coverage of the socio-technical systems of smart cities. The increasing adoption of digitalization and the existing smart city applications have pointed out several challenges in building such IoT frameworks in cities, ranging from security and network reliability to data management aspects. While there have been many approaches proposed to deal with these challenges, a suitable collaboration scheme between the existing IoT solutions, heterogeneous devices and services remains as an open issue [5].

ISCO Project's approach towards more integrated solutions is to design and develop open platforms and tools for interconnecting heterogeneous entities in a city through what we call *smart city objects (SCOs)*. While the project also covers the networking and security aspects for the interoperability and accessibility of all city objects, our focus in this paper remains on the three core modules of ISCO – represented in Fig. 1 by the SCO API, Service Directory, and

[1] http://www.gt-arc.com/projects/isco/.

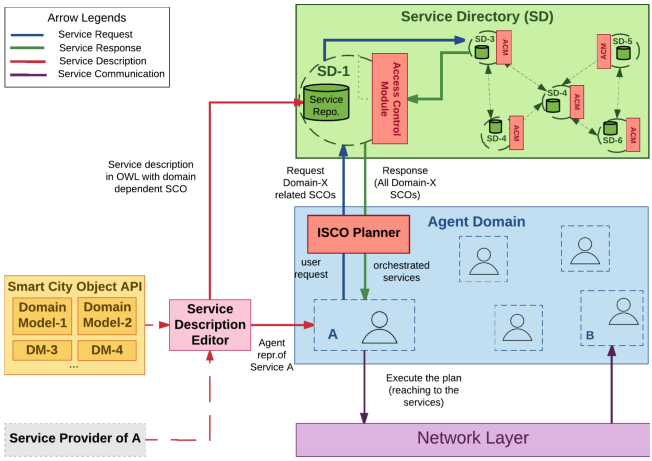


Fig. 1. ISCO architecture: overall interaction of main components

Agent Domain components. In the high-level service workflow of ISCO depicted in Fig. 1, service providers can introduce their service descriptions using our relevant domain models, after which a software agent representation of the service is automatically created under the ISCO agent domain. The description is also saved within the Service Directory (SD) in the Web Ontology Language (OWL) format. The SD is structured as a dynamic collection of distributed nodes and serves as the repositories and request points for registered services. Every service agent contains an instance of the ISCO Planner to request and orchestrate services. Once a user makes a request, the agent looks up domain-related SCO descriptions distributed in SD infrastructure using an Information-Centric Network (ICN) overlay, as described in Sect. 4. When a set of matching services is returned, the access control module filters out services that are not authorized for the requester. ISCO Planner can now process the services first to filter based on their qualities then to find an applicable orchestration of the services based on the user request. We present the details on the inner-workings and the interactions of these components in the remainder of the paper.

3 Semantic Description of Smart City Objects

For sharing knowledge and modeling SCOs in a huge heterogeneous and dynamic environment, the usage of ontologies is the de-facto approach; domain and data models described in an ontology provide a sophisticated semantic description that can be parsed by an application. Existing approaches that model the IoT domain in a descriptive way commonly adopt the IoT-Lite ontology [6] utilized in FIESTA-IoT[2]. However, this lightweight approach lacks a functional description that can be called by a requester.

[2] http://fiesta-iot.eu/.

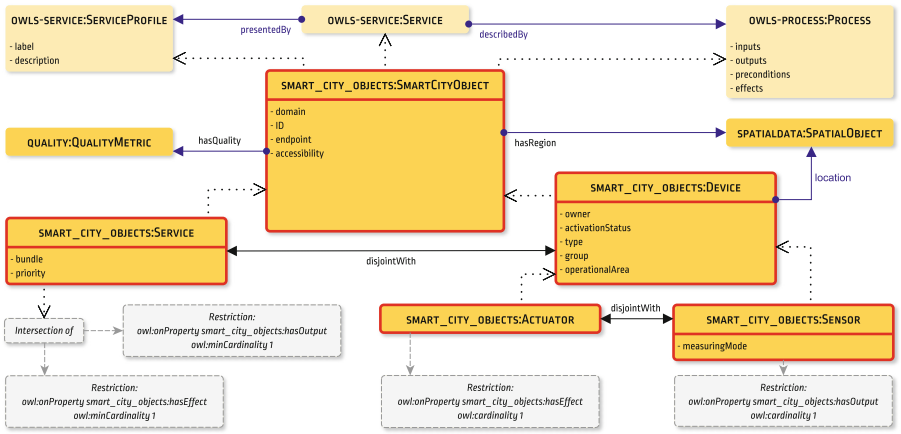


Fig. 2. Centralized overview of the ISCO Smart City Object ontology

On the other hand, for the service-oriented computing community developed well-established standards to describe and invoke functionalities, such as WSDL/SOAP and REST services, but these do not suffice a smart city. Semantic descriptions of software services provide the needed additional information layer, for which several approaches, such as OWL-S [7], WSMO and SAWSDL, have been proposed in recent years, which extend typical service information by preconditions and effects, and define input and output information in a more expressive ontology language, such as OWL. However, these semantic approaches only address web services and not the smart city domain in particular. Our approach extends the OWL-S ontology for smart cities by concepts of IoT-Lite with focus on orchestration. The central class is the abstract *SmartCityObject* class, which is a subclass of the main OWL-S classes; SCO instances are either virtual services or devices, which are further classified into *Actuators* and *Sensors*. The non-functional attributes such as context [8], QoS [9], associated sub domains [10] as well as management attributes enable for sophisticated filtering mechanisms in a smart city with its large amount of functionally overlapping SCOs. An overview of our ontology can be seen in Fig. 2.

4 Scalable Service Lookup and Discovery

Due to heterogeneity and cross domain requirements, smart city IoT solutions rely on an object discovery infrastructure to provide descriptions about their attributes, location, and access methods, among others. Depending on the application, discovery can be a stand-alone service or integrated with the entity management and gateway functions of an IoT middleware. Nevertheless, it aims at providing scalable services for object registration, mapping, and lookup. As such, object descriptions can be distributed based on logical domain, geographical location, or platform specific hierarchy:

- *Domain-specific discovery* builds on the Domain Name System (DNS) of the Internet and is adopted by some projects such as IoT6 by leveraging IPv6 and proposing service discovery (DNS-SD)[3] and mDNS[4] discovery protocols. Global object clusters are discovered with DNS-SD based on IPv6 and higher level protocols CoAP [11]. In local groups, the multicast mDNS is employed for automatic registration and discovery of devices. The Object Name Service (ONS) [12] used in SmartAgriFood is a similar service to DNS for discovery and resolve physical object with EPC code. A local ONS server looks up product descriptions for scanned code by mapping it to a set of resource descriptions provided by external services. SmartAgriFood employs ONS for discovery of entities in production chains in conjunction with a decentralized semantic storage for object data.
- *Geolocation based discovery* is common in device centric and location based applications. The objects are addressed based on their notation of geographic points, areas, or network cluster. While the indexing and geo-discovery are straightforward, additional resolution infrastructure is still required to provide operational details of the resources, as in IoT-A and BUTLER projects.
- *Service Directory (SD)* Whether domain or geographical discovery are used, a directory structure holding rich descriptions of IoT entities is often required in addition to previous distribution approaches. Beside accessibility description, attributes about the entities and their relationships provide data needed for, among others, management, service composition logic of the applications. Semantic web approach is adopted by the projects and is referred as semantic discovery. OWL-based ontologies capture models of physical, logical entities and their relationships, e.g., device capabilities, clustering, QoS requirements. The semantic descriptions allow discovery and matching of services or devices at runtime using SPARQL queries. To meet the required scalability, directory services consist of distributed servers (nodes), which are organized as a multi-level hierarchy or peer-to-peer topology. In contrast, we propose a scalable service directory infrastructure, which features a flat, self-organized topology of distributed service directory nodes.

ISCO Service Directory Based on an ICN Overlay. We propose an IoT service directory (SD) in ISCO that self-organizes the distributed storage and retrieval of smart object descriptions. Its flat architecture makes the directory eligible for universal service discovery for IoT by removing the dependency on discovery mechanism from specific applications and domains.

The underlying principle of ICN is that a communication network should allow a content consumer to focus on the data it needs, named content, rather than having to reference a specific, physical location where that data is to be retrieved from, i.e., named hosts, as in current Internet architecture. Both types of packets carry a name that identifies a piece of data. The consumer sends an INT with the name of the data it needs. When an intermediate node receives the

[3] http://www.dns-sd.org/.
[4] http://www.multicastdns.org/.

INT, it looks for the data in its content store (CS). If the data is not found, it forwards the INT to the next nodes and keeps track of the incoming and outgoing network interfaces for this data in pending interest table (PIT). A series of such forwarding actions creates a breadcrumb path the INT has passed. When the INT arrives at the source node, the requested data is put into DATA and sent back the path towards the consumer. A previous forwarding node receives the DATA, it removes the data name entry in PIT table and adds an entry with the name and network interfaces to forward the data to consumers in the FIT table. Intermediary nodes on the path cache the DATA in their CSs for subsequent INTs. ICN offers a wide range of benefits, e.g, content caching, simpler configuration of network devices, and security at the data level. The distributed directory solution takes advantage of the ICN features to design refined SD functionalities, i.e., developing attribute-based object query methods and content caching strategies for reduced storage overhead as well as increased responsiveness and accuracy.

ICN Based Naming Scheme for City Objects. Each SD-Node acts as an ICN router, which serves the requests for SCO's description by its name, or forwards the requests towards other nodes holding the description. Therefore, the design of a naming scheme affects the performance of objects discovery. ICN naming adopts the semantics of Universal Resource Identifier (URI) scheme. However, the host part does not imply location of the resource, but rather identifies its owner or search domain. The *parameters* part enables the expression of SCO attributes that can be used to look up and discover the SCOs regardless of where they are stored. As shown in Fig. 3, an ICN name of a sensor contains rich semantics describing its domain, location, type, etc. Matching of query attributes and the descriptions is handled by a semantic matcher component in each SD-Node. Depending on the use-case, a strategy to store descriptions and to forward the requests based on object attributes can be dynamically configured. Additionally, various caching and forwarding strategies can be designed to best serve the query demands and SD infrastructure performance.

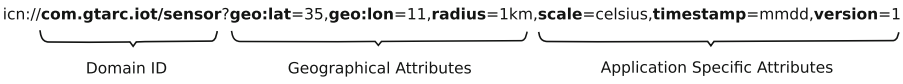


Fig. 3. IoT resource naming scheme in ICN based service directory

Service Directory Node Architecture. The ISCO service directory is constituted by a distributed collection of SD nodes as depicted in Fig. 4. The design of an individual SD-Node, which contains semantic descriptions of SCOs, is shown in Fig. 5. Each functionality is implemented as a modular component: *(1) Triple Store (TDB)* is a component of the Jena[5] project, which serves as a

[5] https://jena.apache.org/documentation/tdb/index.html.

high performance RDF store and query of SCO attributes. *(2) Matcher* component implements mapping methods between requested search attributes and suitable SCO descriptions, which are potential search results. *(3) ICN Router* enables connectivity between SD-Nodes with ICN transport protocol to form a distributed SD infrastructure. Discovery of SCO descriptions is realized by the exchange of interest messages with SCO names. *(4) Query Interfaces* provide distributed application protocols, which allow higher level services to access SCO descriptions and ontologies. It employs various application transport protocols, such as CoAP, MQTT and Rest.

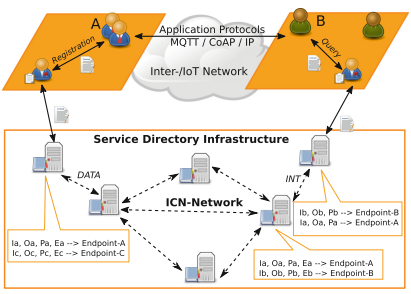


Fig. 4. ICN-based service directory

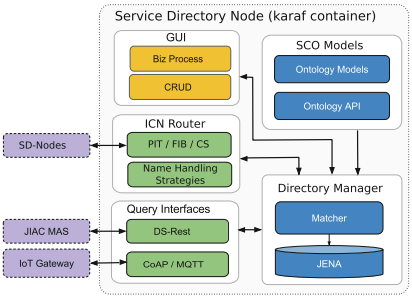


Fig. 5. SD-Node architecture

Scalability. An ICN architecture, in contrast to a host-centric one, does not dictate a predefined hierarchy, e.g., conformance with IP routing or a specific discovery protocol (DNS), among others. This results in a flat network with self-organized topologies. The attribute-based discovery only depends on how an approach describes the devices and services, specifically, their semantic models, matching approaches, and strategies for information organization. Figure 4 illustrates a distributed SD infrastructure utilized by the ISCO platform based on a multi-agent system architecture. Forwarding and caching strategies can be adapted; e.g., the choice of lifetime of the replicas implies a trade-off between the dissemination of descriptions closer to requesting agents, and timeliness, consistency of the information. Moreover, the self-organizing topology allows additional SD-nodes to be added or removed on-demand by applying cloud computing or container technologies (e.g., Docker).

5 Service Composition and Planning

The ISCO middleware planning layer is responsible for generating a service composition, involving different SCOs, that satisfies the specified – functional and qualitative – application requirements at runtime. The adaptability of this module is crucial, as it has to suit different contexts and serves a wide range of applications with varying goals. We are therefore applying the DYNAMICO [13]

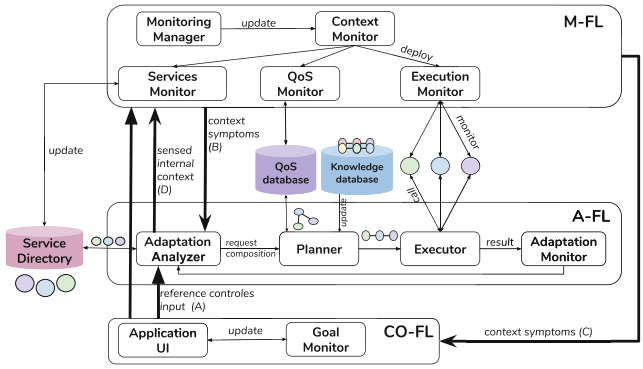


Fig. 6. An overview of the ISCO planning layer components

design guidelines for adaptive systems. This reference model defines three levels of dynamics: the *Objective Feedback Loop (CO-FL)*, the *Adaption Feedback Loop (A-FL)*, and the *Monitoring Feedback Loop (M-FL)*:

Objective Feedback Loop. In dynamic software systems, the adaptation goals (or objectives) should be defined and monitored. These control objectives can define functional system requirements or refer to non-functional system properties (e.g., QoS). The monitoring of these adaptation goals needs an explicit formalization, which e.g., is accomplished in AI planning through a goal state. The goal state contains the facts the system should achieve. In ISCO, we are using an IOPE (input, output, precondition and effect) representation to define the functional goals (e.g., plan a trip) and QoS to state the non-functional system requirements (e.g., trip cost and duration). During the execution of software systems in dynamic environments, the adaptation might be affected by several changes (e.g., goal change, goal is no longer reachable, goal order change, etc.). The system should therefore monitor and evaluate its goals to select the appropriate adaptation.

Adaptation Feedback Loop. (A-FL) receives the adaptation goals from the CO-FL and the monitored context information from the M-FL and selects the appropriate adaptation mechanism to maintain or reach the system goals. Our A-FL layer implements different approaches that enable the system to adapt to system-wide changes. In this loop, the adaptation process might be initiated due to changing control objectives or context information. The A-FL has four main components introduced: **(1) Planner** is responsible for combining SCOs by connecting their IOPEs to generate new composite services that satisfy the client application requirements. We extend the traditional QoS-aware WSC to support IoT devices and sensors. The generated plans should fulfill the functional and qualitative system requirements defined by the CO-LP. **(2) Interpreter** is responsible for the execution of the composite services. The execution is monitored whereas the current state is forwarded to the analyzer module. This process may fail or the results may deviate from the specified goals. In

this case, the adaptation analyzer should trace those deviations and replace the missing or misbehaving components to maintain the system robustness. **(3) Adaptation Analyzer** evaluates the current adaptation goals, selects the most suitable adaptation mechanism and initiates the adaptation process. It also identifies and deploys the required monitoring modules. This module stores, for each new request, the generated service composition along with its global QoS in the knowledge database (KDB). **(4) Adaptation Monitor** checks the state of the adaption mechanism. The selected adaption needs to change if it is no longer adequate for the current system state. This monitoring is done to observe the performance of the adaption mechanisms.

Monitoring Feedback Loop. Self-adaptive systems need to maintain their context-awareness relevance, in order to adapt at runtime to changing context. The M-FL is the context manager in the DYNAMICO reference model (see Fig. 6). The M-FL deploys different context gathers, which monitors the current system context, and reports updates to the A-FL. Our ISCO platform implements four different monitoring components each of which is targeting a specific system component or process: **QoS Monitor** is responsible for monitoring the QoS parameters of the supported services and devices. The measured QoS at runtime may deviate from the defined SLA used to generate a service composition, and should therefore be updated. **Services Monitor** Service developers are able to create new services or update the functional requirements of their services. These updates have to be considered during the planning phase in order to guarantee the correctness of the final composition. This component observes the service directory and notifies changes to the adaptation analyzer. **Execution Monitor** – During the execution of a composite service several issues may arise (e.g., timeout exception, network exception, the returned values does not have the right format). This module reports the tracked issues to the adaptation analyzer, which then initiates the most appropriate recovery process, e.g., replacing unavailable services with similar ones or generating a new sub compositions. **Context Monitor** captures changes in the context of the adaption system. This monitoring is done to be able to adapt to changing conditions in the environment, e.g., changing legal rules of the planning domain.

6 Summary and Future Work

We presented the concept of our ISCO framework, a holistic approach for service discovery and composition in smart cities. The current effort focuses on providing an ecosystem that eases the implementation and deployment of dynamic and self-adaptive software. Our ongoing work tackles the common challenges IoT projects face, which are mainly the lack of integrity and interoperability of cross-domain platforms. The unified SCO, the service directory and the service composition and planning layers are the main components of this framework. While our ongoing work focuses on the final stages of development and integration of these components, our planned future work includes the testing of adaptation capabilities and scalability of the system in a heterogeneous dynamic testbed with both physical and simulated entities.

Acknowledgment. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) under the grant number 16KIS0580.

References

1. Bibri, S.E., Krogstie, J.: ICT of the new wave of computing for sustainable urban forms: their big data and context-aware augmented typologies and design concepts. Sustain. Cities Soc. **32**(Suppl C), 449–474 (2017)
2. Gharaibeh, A., Salahuddin, M.A., Hussini, S.J., Khreishah, A., Khalil, I., Guizani, M., Al-Fuqaha, A.: Smart cities: a survey on data management, security, and enabling technologies. IEEE Commun. Surv. Tutor. **19**(4), 2456–2501 (2017)
3. Brandt, T., Donnellan, B., Ketter, W., Watson, R.T.: Information systems and smarter cities: towards an integrative framework and a research agenda for the discipline. In: AIS Pre-ICIS Workshop-ISCA 2016 (2016)
4. Manville, C., Cochrane, G., Cave, J., Millard, J., Pederson, J.K., Thaarup, R.K., Liebe, A., Wissner, M., Massink, R., Kotterink, B.: Mapping Smart Cities in the EU. EU Publications Office, Luxemburg (2014)
5. Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., Siano, P.: IoT-based smart cities: a survey. In: 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), pp. 1–6 June 2016
6. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-Lite: a lightweight semantic model for the internet of things. In: IEEE International Conference on Ubiquitous Intelligence and Computing, pp. 90–97. IEEE (2016)
7. Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: semantic markup for web services, November 2004
8. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
9. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. (CSUR) **46**(1), 1 (2013)
10. Neirotti, P., De Marco, A., Cagliano, A.C., Mangano, G., Scorrano, F.: Current trends in smart city initiatives: some stylised facts. Cities **38**, 25–36 (2014)
11. Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252, June 2014
12. GS1: GS1 Object Name Service (ONS) Version 2.0.1. Ratified Standard 2 (2013)
13. Villegas, N.M., Tamura, G., Müller, H.A., Duchien, L., Casallas, R.: DYNAMICO: a reference model for governing control objectives and context relevance in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II. LNCS, vol. 7475, pp. 265–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_11

CJM-ab: Abstracting Customer Journey Maps Using Process Mining

Gaël Bernard[1(✉)] and Periklis Andritsos[2]

[1] Faculty of Business and Economics (HEC), University of Lausanne, Lausanne, Switzerland
gael.bernard@unil.ch

[2] Faculty of Information, University of Toronto, Toronto, Canada
periklis.andritsos@utoronto.ca

Abstract. Customer journey mapping (CJM) is a popular technique used to increase a company's understanding of their customers. In its simplest form, a CJM shows the main customer paths. When dealing with complex customers' trajectories, these paths are difficult to apprehend, losing the benefit of using a CJM. We present a javascript-based tool that can leverage process mining models, namely process trees, and business owners' knowledge to semi-automatically build a CJM at different levels of granularity. We applied our approach with a dataset describing a complex process, and shows that our technique can abstract it in a meaningful way. By doing so, we contribute by showing how process mining and CJM can be put closer together.

Keywords: Customer journey mapping · Process mining
Process tree · Customer journey analytics

1 Introduction

A customer journey map (CJM) is a conceptual tool used to visualize typical customers' trajectories when using a service. In their simplest form, CJMs show the interactions between a customer and a service provider through time. A series of interactions is called a journey. Because CJMs give a company a better understanding of their customers, they are becoming increasingly popular amongst practitioners. A CJM can be used as a design thinking tool by internal stakeholders to anticipate the best – or worst – journeys possible. Such journeys, displayed on a CJM, are called the *expected journeys*. However, customers might experience a different journey from the one anticipated. For this reason, few researchers [4,5,10] propose leveraging traces left by customers in information systems to build CJMs from evidence. Because the journeys that will be displayed on the CJM are produced from facts, we refer to them as the *actual journeys*. Such approaches are in line with the urgent call from the authors Lemon and Verhoef to take a data-driven approach to map the customer journey [15].

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 49–56, 2018.
https://doi.org/10.1007/978-3-319-92901-9_5

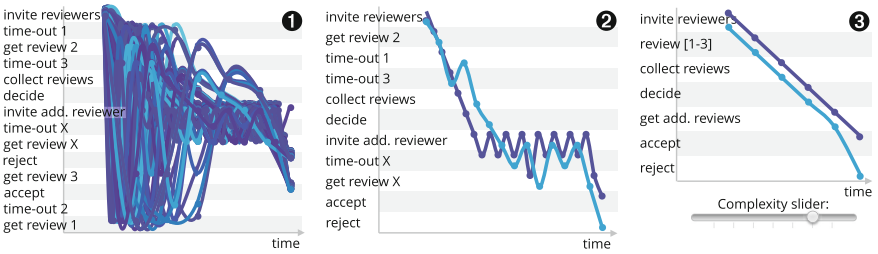


Fig. 1. Three possible ways to display the handling of reviews for a journal from [1] on a CJM: ❶ projecting the actual journeys (only the first 100 – out of 10,000 – journeys are displayed); ❷ using two representative journeys; and, ❸ using two representative journeys and abstracting the activities using the technique presented in this paper.

However, when dealing with numerous journeys, it becomes unrealistic to display all the actual journeys on a single CJM. For illustration purposes, Fig. 1 depicts 10,000 instances of the traces related to the handling of reviews for a journal, a synthetic dataset available in [1]. In the context of this dataset, *the service provider* is the conference's organizing committee, *the customers* are the researchers submitting their papers, and *a journey* describes the handling of the reviews, from the submission until the final decision. In Fig. 1, part ❶, it is difficult to apprehend the typical paths of the reviewing process. To this end, *representative journeys* have been introduced as a means of reducing the complexity. Indeed, the central CJM (❷) uses two representative journeys to summarize 10,000 actual journeys. Although representative journeys decrease the complexity by reducing the number of journeys, a CJM might still be difficult to apprehend when it is composed of many activities. Indeed, even though only representative journeys are used, quickly spotting the main differences between the two journeys visible in ❷ (Fig. 1) is not straightforward due to the high number of activities and the length of the journeys.

We propose CJM-ab (for CJM abstractor) a solution that leverages the expertise of process discovery algorithms from the process mining discipline to abstract CJMs. More precisely, we take as an input a process tree, we parse it, starting from the leaves, and iteratively ask the end-user if it is relevant to merge the activities that belong to the same control-flow, and, if so, to provide a name for this group of activities. By doing so, we let the end-user decide which activities should be merged and how they should be renamed. Then, one can visualize the same CJMs at different levels of granularity using a slider, which is visible in Fig. 1, part ❸. At a certain level of granularity, we clearly observe, given the end activities, that one representative journey summarizes the accepted papers, while the other one depicts the rejected papers. The importance and originality of CJM-ab is that it explores, for the first time, a seamless integration of business process models with customer journeys maps.

The paper is organized as follows. Section 2 introduces process mining and the process discovery activity. Section 2.2 describes the customer journey discovery activity. Section 3 describes our algorithm, and Sect. 4 provides a demonstration. Finally, Sect. 5 opens a discussion and concludes the paper.

2 Background

2.1 Process Mining and Process Discovery

Our approach is a seamless integration of Process Mining with Customer Journey Mapping and showcases the impact that the latter can have in the analysis of journeys. Process mining is an emerging discipline sitting between machine learning and data mining on the one hand, and process modeling and analysis on the other [2]. In this research, we focus on the discovery of process models, one of the three types of process mining along with conformance and enhancement.

The idea behind the discovery of process models is to leverage the evidence left in information systems to build process models from event logs. The resulting process models are, therefore, based on factual data, showing how the process was really executed. To build such a model, process mining uses an input data format called event logs. An event log is a collection of traces, a trace being a single execution of a process composed of one or multiple activities.

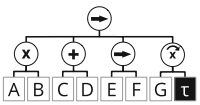


Fig. 2. One of the possible process trees given the event log $T = (\langle \text{BDCEF} \rangle, \langle \text{ACDEFG} \rangle, \langle \text{BCDEFGG} \rangle)$

For illustration purposes, let $T = (\langle \text{BDCEF} \rangle, \langle \text{ACDEFG} \rangle, \langle \text{BCDEFGG} \rangle)$ be an event log composed of 3 traces and 7 distinct activities. Regardless of the notation, the resulting models can express the control-flow relations between activities. For instance, for the event log, T, the model might express the following notation: (1) A and B are in an XOR relation ($\times$); i.e., only one of them is executed; (2) C and D are executed in parallel ($+$); i.e., both activities are executed in any order; (3) E and F are in a sequence relation ($\rightarrow$); i.e., F always follows E; (5) G is in a XOR loop (Combination of $\times$ and $\circlearrowleft$); i.e., it can be executed 0 or many times. Note that τ denotes a silent activity. It is used to correctly execute the process but it will not result in an activity which will be visible in the event logs. Figure 2 displays the five aforementioned relations using a process tree.

Discovering a process model from event logs is a challenge. Indeed, state-of-the-art algorithms should be robust enough to generalize (to avoid overfitting models) without being too generic. They should also try to build process models that are as simple as possible [3]. Many representations exist to express the discovered process models: Petri nets, YAWL, process trees, state machines, or bpmn models, to name a few. The next section introduces the notation used by our algorithm: process trees.

Process Tree. A process tree is an abstract hierarchical representation of a process model introduced by Vanhatalo et al. [17], where the leaves are annotated with activities and all the other nodes are annotated with operators such as $\times$ [14]. One interesting characteristic of process trees is that they guarantee the soundness of the models. A model is considered to be not sound when some activities cannot be executed or when the end of the process cannot be reached. The soundness guarantee is one reason that we choose the process tree notation. There are also three other reasons. First, process models in block structure

achieve best performance in terms of fitness, precision, and complexity [3]. Second, the hierarchical structure of process trees is ideal to derive multiple levels of granularity. Finally, according to Augusto et al. [3], process trees are used by top-performing process model algorithms, such as the inductive miner [11–13] or the Evolutionary Tree Miner [6].

2.2 Customer Journey Discovery

In [5], we proposed a process mining based model that allows us to map a standard event log from process mining (i.e., XES [9]) to store customer journeys, a first attempt to bring customer journeys and process mining closer together.

Discovering a set of representative journeys that best describe the actual journeys observed in the event logs is a challenge inspired by the process discovery challenge introduced in the previous section. However, instead of describing the control flows of activities using a business process model, the main trajectories (i.e., the representative journeys) are shown using a CJM. It encompasses three important challenges: (1) choosing the number of representatives. Let k be this number of representative journeys used on a CJM; (2) grouping actual journeys in k clusters; and (3) for each k, finding a representative journey. We briefly present these three challenges and ways to overcome them.

The first challenge is to set the number of representative journeys used to summarize the entire actual journeys. Looking at ❶ from Fig. 1, it is difficult to say how many representative journeys should be used to summarize the data. We identify two ways to solve this challenge. The number of representative journeys can be set manually, or it can also be set using standard model selection techniques such as the Bayesian Information Criterion (BIC) penalty [16], or the Calinski-Harabasz index [7].

Once k has been defined, actual journeys should be split in k clusters and a representative journey per cluster must be found. One of the ways, presented in [4], is to first define a distance function between actual journeys, such as the edit distance, or shingles, and to build a distance matrix; then, to split the actual journeys in k groups using hierarchical clustering techniques. Next, the representative can be found using a frequent sequence mining algorithm [4], by counting the density of sequences in the neighborhood of each candidate sequence [8], by taking the most frequent sequences [8], or by taking the medoid [8]. Instead of inferring the representative from the distance matrix, it is also possible to obtain it using statistical modeling [8]. We can employ an Expectation-Maximization algorithm on a mixture of k Markov models, and then for each Markov model the journey with the highest probability becomes the representative [10].

The next section describes a novel way to leverage business process models to abstract customer journey maps.

3 Abstracting Customer Trajectories Using Process Trees

CJM-ab uses four steps to render a CJM at different levels of abstraction. They are depicted in Fig. 3. This chapter introduces each step. In the first step, the

goal is to build a process tree given an event log. This can be done using the technique introduced in Sect. 2.1. Next, using the same event log, the goal is to build a CJM using the technique introduced in Sect. 2.2.

Fig. 3. Rendering a CJM at different levels of abstraction in four steps

The third step consists of parsing the tree obtained in step 1. To this aim, we developed a script in javascript which parses the process tree (i.e., XML file) and performs a reverse Breadth-first search; i.e., traversing the operators in the tree from the lowest ones to the root in a level-wise way. Let ℓ be the number of operators in the process tree. At each of the ℓ operators of the process tree, we offer the opportunity to the end-user to merge the leaves under the operator. If the user chooses to merge the activities, she should provide a new name and the operator virtually becomes a leaf. If the end-user chooses not to merge the activities, we keep the leaves intact. If the answer is no, we keep the activities separated at all levels of granularities, and we also disable the parents' steps. Indeed, we postulate that if a user does not want to merge two activities at a low level of granularity, it does not make sense to merge them later at a higher level of granularity.

Input : cjm, customer journey map
λ, level of abstraction
pt, process tree annotated with merging decisions
Output: cjm_λ, cjm at the level of abstraction λ

```
Function GetLevelAbstraction(cjm, λ, pt)
    for i ← 0 to λ do
        cjm → Abstract(cjm, pt.operator_i)
    return cjm
Function Abstract(cjm, op)
    foreach journey in cjm do
        journey.replace(op.leaves, op.new_name, removeSeqRepeats=True )
    return cjm
```

Algorithm 1. Function to get to the level of complexity λ

Finally, in step 4, we transform the CJM at different levels of abstraction. Let λ be the number of abstractions which will be available for a CJM. It can be

seen as the number of steps that will be included in the sliders visible in Fig. 1, part ❸. Note that λ is equal to the number of times the end-user decides to merge the activities and that $\lambda = \ell$ when the end-user merges all the activities. Let $operator_\lambda$ be the λ^{th} operator to be merged. Let GetLevelAbstraction(cjm, λ, pt) be a function that returns a CJM at the λ^{th} level of abstraction. Algorithm 1 shows how the function Abstract is recursively called to get to the level of abstraction λ. The parameter removeSeqRepeats in Algorithm 1 in line 7 emphasizes that continuous sequence of activities that are to be replaced, will be replaced by only one instance of the new name given for this operator. For instance, if the journey is "AABCBAC", the leaves that are to be replaced, are "A" and "B" and the new name is "X", the journey will become "XCXC". This reduces the length of the journeys and, thus, increases the abstraction. One can go back from more abstract to fine granular again by calling GetLevelAbstraction() again with a smaller λ. The next section illustrates these four steps with a running example.

4 Demonstration

This section provides a running example of our developed tool. The running example is based on synthetic event logs describing the handling of reviews for a journal (from [1]) cited in the introduction. It contains 10,000 journeys and 236,360 activities. This demonstration is available on http://customer-journey.unil.ch/cjm-ab/. In the first step, we obtained a process tree by using the inductive miner [14] with default parameters[1]. It results in the process tree visible in Fig. 4. In the second step, we obtain a CJM by: (1) measuring the distance between actual journeys using the edit distance; (2) building a dendrogram using a hierarchical clustering algorithm; (3) finding k using the Calinski-Harabaz Score ($k = 2$); (4) finding representative journeys using the function 'seqrep' available in Traminer, a R package[2]. It results in a CJM which is visible in

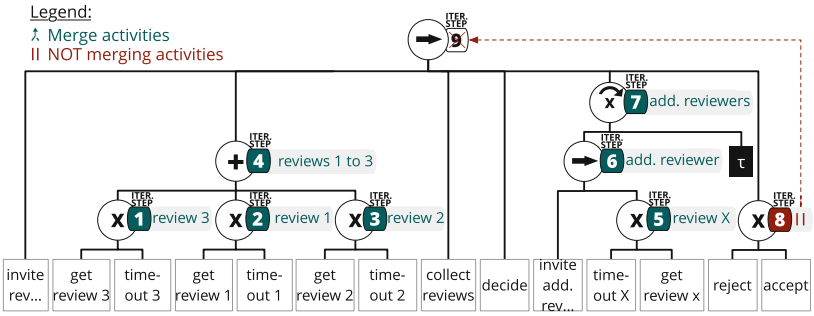


Fig. 4. Process tree annotated with the order in which the operators are parsed (i.e., 'iter. step') and the decisions to merge the activities or not (i.e., colors red and green).

[1] Using the software ProM available at http://www.promtools.org/doku.php.
[2] Available at: http://traminer.unige.ch/doc/seqrep.html.

❷ (Fig. 1). In the third step, we parse the XML in javascript. To traverse the tree, we are using a tree-like data structures [3]. The order in which the operators are parsed is depicted in Fig. 4 (i.e., 'step'). Figure 4 shows that we decided to merge 7 out of the 9 operators (in green in Fig. 4). Note that we decided not to merge the activities 'reject' and 'accept', which disabled the option of merging all the activities below step 9. The Fig. 5 shows a screen capture of the application when merging the activities during step 1. Finally, the Fig. 6 shows the resulting CJMs at three levels of abstraction.

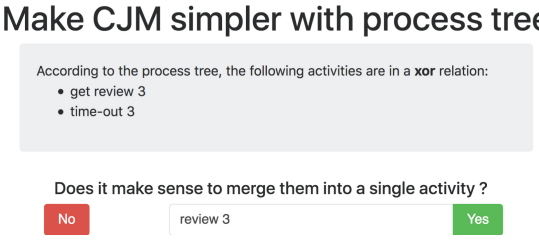


Fig. 5. Screen shot of the application during the merging process at 'step 1'

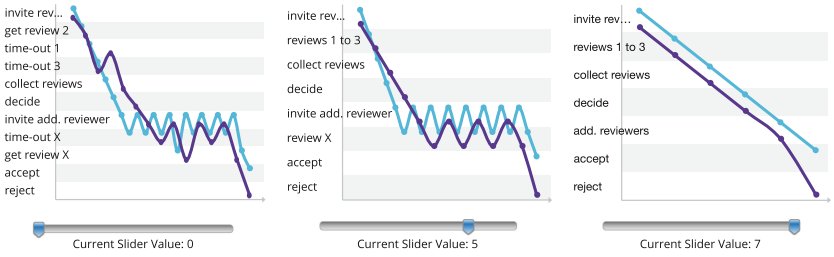


Fig. 6. Results at the levels of abstraction 1, 3, and 7.

5 Conclusion

CJMs are being used more and more to help service providers put themselves in their customers' shoes. However, very little research has investigated automated ways of building them. We contribute by showing how a process mining model can be used to guide the abstraction of a CJM. By answering few questions about the merging of the activities and by playing with the abstraction sliders, we anticipate that our tool allows practitioners to gain new insights about their data. By leveraging process trees – a format built within the process mining community – we can bring customer journey analytics and process mining closer together. We expect that many algorithms and works from process mining are relevant for the discovery of customer journeys.

[3] Available at: https://github.com/joaonuno/tree-model-js.

References

1. van der Aalst, W.: Synthetic event logs - review example large.xes.gz (2010). https://doi.org/10.4121/uuid:da6aafef-5a86-4769-acf3-04e8ae5ab4fe
2. van der Aalst, W.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
3. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. arXiv preprint arXiv:1705.02288 (2017)
4. Bernard, G., Andritsos, P.: CJM-ex: goal-oriented exploration of customer journey maps using event logs and data analytics. In: 15th International Conference on Business Process Management (BPM 2017) (2017)
5. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: Proceedings of the Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017) (2017)
6. Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int. J. Coop. Inf. Syst. **23**(01), 1440001 (2014)
7. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat.-Theory Methods **3**(1), 1–27 (1974)
8. Gabadinho, A., Ritschard, G., Studer, M., Müller, N.S.: Extracting and rendering representative sequences. In: Fred, A., Dietz, J.L.G., Liu, K., Filipe, J. (eds.) IC3K 2009. CCIS, vol. 128, pp. 94–106. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19032-2_7
9. Günther, C.W., Verbeek, E.: XES-standard definition (2014)
10. Harbich, M., Bernard, G., Berkes, P., Garbinato, B., Andritsos, P.: Discovering customer journey maps using a mixture of Markov models, December 2017
11. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
12. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
13. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 91–110. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_6
14. Leemans, S.: Robust process mining with guarantees. Ph.D. thesis, Eindhoven University of Technology (2017)
15. Lemon, K.N., Verhoef, P.C.: Understanding customer experience throughout the customer journey. J. Mark. **80**(6), 69–96 (2016)
16. Schwarz, G., et al.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
17. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_10

PRESISTANT: Data Pre-processing Assistant

Besim Bilalli[1,2(✉)], Alberto Abelló[1], Tomàs Aluja-Banet[1], Rana Faisal Munir[1], and Robert Wrembel[2]

[1] Universitat Politècnica de Catalunya, Barcelona, Spain
{bbilalli,aabello,fmunir}@essi.upc.edu, tomas.aluja@upc.edu
[2] Poznan University of Technology, Poznan, Poland
robert.wrembel@cs.put.poznan.pl

Abstract. A concrete classification algorithm may perform differently on datasets with different characteristics, e.g., it might perform better on a dataset with continuous attributes rather than with categorical attributes, or the other way around. Typically, in order to improve the results, datasets need to be pre-processed. Taking into account all the possible pre-processing operators, there exists a staggeringly large number of alternatives and non-experienced users become overwhelmed. Trial and error is not feasible in the presence of big amounts of data. We developed a method and tool—PRESISTANT, with the aim of answering the need for user assistance during data pre-processing. Leveraging ideas from meta-learning, PRESISTANT is capable of assisting the user by recommending pre-processing operators that ultimately improve the classification performance. The user selects a classification algorithm, from the ones considered, and then PRESISTANT proposes candidate transformations to improve the result of the analysis. In the demonstration, participants will experience, at first hand, how PRESISTANT easily and effectively ranks the pre-processing operators.

Keywords: Data pre-processing · Meta-learning · Data mining

1 Introduction

Although machine learning algorithms have been around since the 1950s, their initial impact has been insignificant. With the increase of data availability and computing power, machine learning tools and algorithms are being included in many Information Systems, and are making breakthroughs in very diverse areas. Their success has raised the need for mainstreaming the use of machine learning, that is, engaging even non-expert users to perform data analytics. However, the multiple steps involved in the data analytics process render this process challenging.

Data analytics, sometimes referred to as knowledge discovery [8], consists of *data selection*, *data pre-processing*, *data mining*, and *evaluation* or *interpretation*.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 57–65, 2018.
https://doi.org/10.1007/978-3-319-92901-9_6

Table 1. Summary of `Automobile`

Metadata	Value
Instances	205
Attributes	26
Classes	2
Categorical attributes	11
Continuous attributes	15
Missing values	59

Table 2. Transformations on `Automobile`

Transformation	Attribute	PA
Unsup. discretization	1, 9, 10, 11, 12, 13	**0.81**
Unsup. discretization	1, 9, 10	**0.80**
Unsup. discretization	All cont. Atts.	**0.75**
Sup. nom. to binary	All cat. atts.	**0.73**
Unsup. normalization	All cont. atts.	**0.71**

A very important and time consuming step that marks itself out of the rest, is the data pre-processing step.

Data pre-processing is challenging, but at the same time has a heavy impact on the overall analysis. Specifically, it can have significant impact on the generalization performance of a classification algorithm [16], where performance is measured in terms of the ratio of correctly classified instances (i.e., predictive accuracy). A brief real-world example can be used to demonstrate this.

Let us suppose that a multinational insurance company has an Information System to be used by different actuaries world wide. These want to apply a classification algorithm (e.g., Logistic Regression) to different *Automobile*[1] datasets like the one, whose summary is given in Table 1. This dataset specifies autos in terms of their various characteristics like *fuel type*, *aspiration*, *num-of-doors*, *engine-size*, etc. The response attribute (i.e., class) is *symboling*. Symboling is a categorical attribute that indicates the insurance risk rate, and it can take the following values: $-3, -2, -1, 0, 1, 2, 3$ (value 3 indicates that the auto is risky, -3 that it is pretty safe). The goal is to build a model that will predict the insurance risk rate for a new auto.

Now, if *Logistic Regression* is applied to the original non-transformed dataset, a predictive accuracy of 0.71 is obtained with a 10 fold cross-validation. In contrast, if some pre-processing were to be applied in advance, the results shown in Table 2 would be obtained, where the first column denotes the transformation applied, the second denotes the index values of the attributes to which the transformation is applied and the third is the predictive accuracy obtained after the Logistic Regression algorithm is applied on the transformed dataset. Note that for instance, if the transformation *Unsupervised Discretization* (with default parametrization) is applied to attributes $\{1, 9, 10, 11, 12, 13\}$, an improvement of 14% is obtained in terms of predictive accuracy. A non-experienced user (at times even an experienced user) cannot be aware of that. His/her only solution is to execute all the possible transformations and then apply the algorithm after each transformation. However, this is generally unfeasible in the presence of big amounts of data, due to the high computational complexity of data mining algorithms. Hence, a proper recommendation of transformations would ease the user's task and at the same time it would improve the final result.

[1] https://archive.ics.uci.edu/ml/support/Automobile.

Table 3. List of transformations (data pre-processing operators)

Transformation	Technique	Attributes	Input type	Output type
Discretization	Supervised	Local	Continuous	Categorical
Discretization	Unsupervised	Local	Continuous	Categorical
Nominal to binary	Supervised	Global	Categorical	Continuous
Nominal to binary	Unsupervised	Local	Categorical	Continuous
Normalization	Unsupervised	Global	Continuous	Continuous
Standardization	Unsupervised	Global	Continuous	Continuous
Replace miss. val.	Unsupervised	Global	Continuous	Continuous
Replace miss. val.	Unsupervised	Global	Categorical	Categorical
Principal components	Unsupervised	Global	Continuous	Continuous

The main tools used for data analytics (e.g., R[2], scikit-learn[3], Weka [11]) overlook data pre-processing when it comes to assisting non-expert users on improving the overall performance of their analysis. These tools are usually meant for professional users—who know exactly which pre-processing operators (transformations) to apply, and they leave non-experts unattended. To remedy this, we developed PRESISTANT[4], which focuses on assisting the users by reducing the number of pre-processing options to a bunch of potentially relevant ones and ranks them. The goal is to highlight the transformations that have higher potential positive impact on the analysis.

PRESISTANT aims at reducing the time consumed in data pre-processing and at the same time improving the final result of the analysis. The focus is on classification problems, thus only operators that improve the performance of a classification algorithm (e.g., increase the predictive accuracy) are recommended.

The remainder of this paper is organized as follows. We first give a brief overview of data pre-processing. Next, we give an overview of PRESISTANT and present its core features. Finally, we outline our on-site presentation.

2 Data Pre-processing

Data pre-processing consumes 50–80% of data analysis time [17]. The reason for this is that it encompasses a broad range of activities. Sometimes data needs to be transformed in order to fit the input requirements of the machine learning algorithm, e.g., if the algorithm accepts only data of numeric type, data is transformed accordingly [14]. Sometimes, data requires to be transformed from one representation to another, e.g., from an image (pixel) representation to a matrix (feature) representation [10], or data may even require to be integrated

[2] https://r-project.org.
[3] http://scikit-learn.org/stable.
[4] For details visit: http://www.essi.upc.edu/~bbilalli/presistant.html.

with other data to be suitable for exploration and analysis [15]. Finally and more importantly, data may need to be transformed with the only goal of improving the performance of a machine learning algorithm [7]. The first two types of transformations are more of a necessity, whereas the latter is more of a choice, and since an abundant number of choices exist, it is time consuming to find the right one. In PRESISTANT, we target the latter type of pre-processing, and as such, the transformations taken into consideration are of the type that can impact the performance of classification algorithms, and they are listed in Table 3. These are the most commonly used transformations in the Weka platform, and their implementations are open source[5].

In Table 3, a transformation is described in terms of: (1) the *Technique* it uses, which can be *Supervised*—the algorithm knows the class of each instance and *Unsupervised*—the algorithm is not aware of the class, (2) the *Attributes* it uses, which can be *Global*—applied to all compatible attributes, and *Local*—applied to specific compatible attributes, (3) the *Input Type*, which denotes the compatible attribute type for a given transformation, which can be *Continuous*—it represents measurements on some continuous scale, or *Categorical*—it represents information about some categorical or discrete characteristics, (4) the *Output Type*, which denotes the type of the attribute after the transformation and it can similarly be *Continuous* or *Categorical*.

3 System Overview

PRESISTANT takes as input a dataset and a classification algorithm, and generates a ranking of transformations according to their predicted impact on the final result of the algorithm. In the following, we explain the method and architecture used to achieve this.

3.1 Meta-Learning for Data Pre-processing

Meta-learning is a general process used for predicting the performance of an algorithm on a given dataset. It is a method that aims at finding relationships between dataset characteristics and data mining algorithms [6]. Given the characteristics of a dataset, a predictive meta-model can be used to foresee the performance of a given data mining algorithm. To this end, meta-learning has shown to be effective in the model-selection problem [1,13] as well as CASH (combined algorithm selection and hyperparameter optimization) problem [9,18].

However, in our previous works [2,4,5], we showed that meta-learning can also be used to provide support specifically in the pre-processing step. This can be done by learning the impact of data pre-processing operators on the final result of the analysis. That is to say, detecting the transformations that after being applied on a dataset, increase the accuracy of a selected classification algorithm. This way, meta-learning pushes the user support to the data pre-processing step

[5] https://github.com/bnjmn/weka.

by enabling a ranking of transformations according to their relevance to the analysis.

Extensive results of the evaluation of this method can be found in [5]. Yet briefly, the evaluation of the rankings with regards to the gain obtained from the user's point of view, using a classical information retrieval metric—Discounted Cumulative Gain (DCG) [12], showed that for the whole set of possible transformations of a dataset, PRESISTANT is as close as 73% to the gain obtained from the ideal rankings, whereas for the best transformation in the ranking, PRESISTANT is as close as 79%, on average for all the considered algorithms [5].

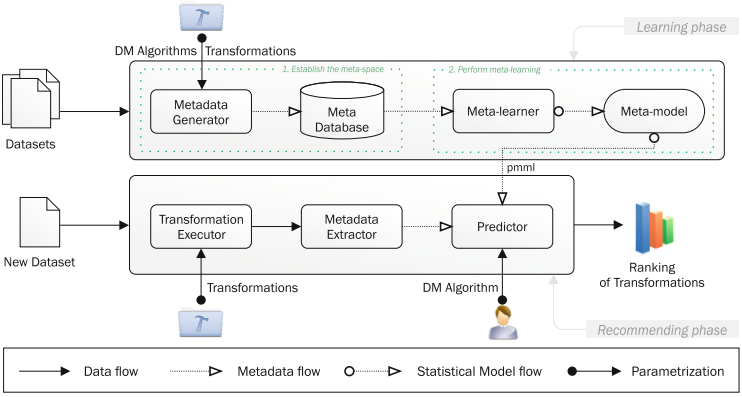


Fig. 1. PRESISTANT: system overview

3.2 Architecture and Implementation

PRESISTANT is built with the meta-learning concept in mind and, as a consequence, it consists of two main phases, the *learning* and the *recommending* phase. Figure 1 depicts the architecture of PRESISTANT.

The *learning* phase is performed offline and consists of two steps. In the first step, a meta-dataset is established for each classification algorithm that is considered by the system for application, i.e., for the time being PRESISTANT supports 5 classification algorithms. The meta-dataset is constructed by extracting dataset characteristics—meta-features, and by generating different measures on the performance of the classification algorithms on the datasets, i.e., predictive accuracy, precision, recall, and area under the roc curve (AUC). Dataset characteristics and performance measure altogether are referred to as *metadata*. In the second step, meta-learning is performed on top of the meta-dataset. As a result, a predictive meta-model is generated that can be used to predict the performance of a classification algorithm on any new dataset.

The *recommending* phase is initiated when a new dataset to be analyzed arrives. At this point, a set of transformations are applied, and the corresponding transformed datasets are obtained. Transformations are applied depending

on whether they are Local or Global (as classified in Table 3). If a transformation is Global it is applied only once to the set of all compatible attributes (e.g., normalizing all numeric attributes), whereas if it is Local, it is applied to: (1) every compatible attribute separately (e.g., discretizing one attribute at a time), and (2) all the set of compatible attributes (e.g., replacing missing values of all attributes). Furthermore, PRESISTANT uses heuristics (cf. [5] for more details), to prune the number of transformations that may be applied. For instance, given that *Standardization* and *Normalization* do not affect the performance of a *Decision Tree*, they are not applied when using a *Decision Tree (J48)* as classifier. Similarly, since in Weka, when applying *Nearest Neighbor (IBk)* the *Normalization* transformation is applied implicitly, PRESISTANT does not check for its impact in an explicit way. Although it may seem computationally costly to execute transformations and then predict the performance of algorithms on the transformed datasets, notice that this is orders of magnitude less costly than applying the classification algorithm after each transformation. This in fact, is the strongest point of our approach.

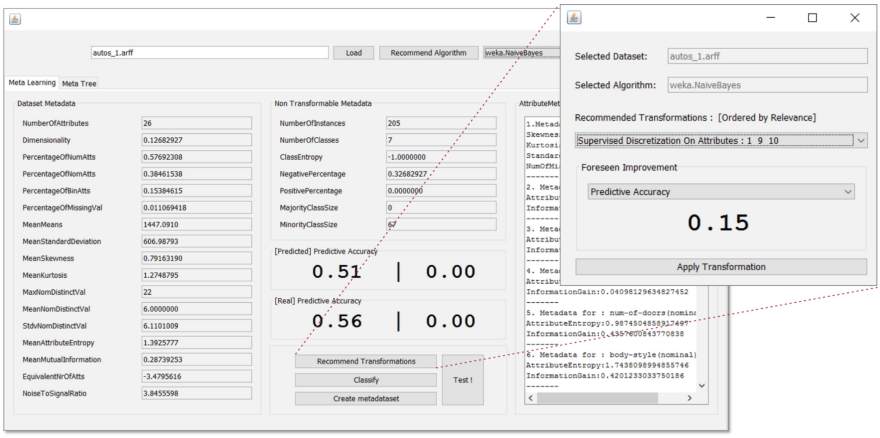


Fig. 2. PRESISTANT: application interface

Furthermore, some concepts that require specific attention are:

Metadata. In our previous work [3], we studied and classified all types of metadata that can be used by systems that intelligently support the user in the different steps of the data analytics process. PRESISTANT, considers: (1) 54 dataset characteristics consisting of different summary characteristics (e.g., number of instances, dimensionality, class entropy, mean attribute entropy, etc.) and (2) a measure of the performance of an algorithm on the dataset (i.e., predictive accuracy, precision, recall or AUC). The metadata generation and extraction are performed in the *Metadata Generator* and *Metadata Extractor* modules, respectively, shown in Fig. 1. These modules are implemented in *Java*.

Meta-learner. The meta-learner is the algorithm that performs learning on top of the meta-dataset constructed out of the above mentioned metadata. The goal of the meta-learner is to create a model that is capable of predicting the performance of a given classification algorithm on a transformed dataset. In other words, its goal is to correctly predict the impact of a transformation on the performance of a classification algorithm. PRESISTANT uses a *Random Forest* as meta-learner. The meta-learning process is performed in the *Meta-Learner* module shown in Fig. 1, and it is implemented in *R*. The generated models are exported into *pmml* files, which are then fed to the *Predictor* module.

Transformations. The set of transformations currently considered in PRESISTANT, that the audience will experience, are described in Table 3 and cover a wide range of data pre-processing tasks, which are distinguished as *data reduction* and *data projection*. *Data reduction* aims at decreasing the size of the dataset (e.g., instance selection or feature extraction). *Data projection*, alters the representation of the data (e.g., mapping continuous values to categories or encoding nominal attributes). The list of transformations considered by PRESISTANT can be extended by adding new ones from the palette of transformations offered in Weka, and then training the meta-models. The execution of transformations is performed in the *Transformations Executor* module, shown in Fig. 1, which is implemented in *Java*.

Classification algorithms. They represent the algorithms for which PRESISTANT can currently provide user support. That is, if the user selects one of these algorithms for analyzing a dataset, PRESISTANT is capable of recommending transformations that will improve the quality of the analysis. PRESISTANT is built on top of Weka, and Weka categorizes the classification algorithms into the following 7 categories: *bayes, functions, lazy, rules, trees, meta-methods and miscellaneous*, out of which the last two contain algorithms that are more complex and almost never used by non-experts. Assistants will be able to experiment with a representative classification algorithm for each category except the last two, and they are: *Naive Bayes, Logistic, IBk, PART, and J48*, respectively.

4 Demo Walkthrough

In the on-site demonstration, the functionalities and capabilities of PRESISTANT (cf. Fig. 2) for assisting the user in the data pre-processing step will be presented. Different real world datasets covering a variety of domains (e.g., health, insurance, banking) will be available to show the benefits of our tool. Demo participants will be encouraged to play the roles of data analysts and validate the tools' assistance. Since, for the time being, PRESISTANT covers 5 classification algorithms, the on-site demonstration will consist of 5 scenarios where each time a different algorithm will be presented. In every scenario, the user will deal with a classification problem, e.g., in the first scenario the user

will be presented with the *lung cancer*[6] dataset where the task will be to build a prediction model that achieves a good accuracy on predicting the type of the lung cancer a patient has, evaluated with 10-fold cross-validation. The idea is to show how the transformations recommended by PRESISTANT improve the quality of the analysis or more precisely how once applied, they increase the predictive accuracy of a chosen classification algorithm. Further datasets for the on-site demonstration will be selected from *OpenML*[7], which contains one of the biggest collection of datasets for classification problems.

Acknowledgments. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC).

References

1. Bilalli, B., Abelló, A., Aluja-Banet, T.: On the predictive power of meta-features in OpenML. Appl. Math. Comput. Sci. **27**(4), 697–712 (2017)
2. Bilalli, B., Abelló, A., Aluja-Banet, T., Wrembel, R.: Automated data pre-processing via meta-learning. In: Bellatreche, L., Pastor, Ó., Almendros Jiménez, J.M., Aït-Ameur, Y. (eds.) MEDI 2016. LNCS, vol. 9893, pp. 194–208. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45547-1_16
3. Bilalli, B., Abelló, A., Aluja-Banet, T., Wrembel, R.: Towards intelligent data analysis: the metadata challenge. In: IOTBD 2016, pp. 331–338 (2016)
4. Bilalli, B., Abelló, A., Aluja-Banet, T., Wrembel, R.: Intelligent assistance for data pre-processing. Comput. Stand. Interfaces **57**, 101–109 (2018)
5. Bilalli, B., Abelló, A., Aluja-Banet, T., Wrembel, R.: PRESISTANT: learning based assistant for data pre-processing. In: eprint arXiv: https://arxiv.org/pdf/1803.01024.pdf (2018). (Under review)
6. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining, 1st edn. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-73263-1
7. Chu, X., Ilyas, I.F., Krishnan, S., Wang, J.: Data cleaning: overview and emerging challenges. In: SIGMOD 2016, pp. 2201–2206 (2016)
8. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Mag. **17**, 37 (1996)
9. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: NIPS 2015, pp. 2962–2970 (2015)
10. Furche, T., Gottlob, G., Libkin, L., Orsi, G., Paton, N.W.: Data wrangling for big data: challenges and opportunities. In: EDBT 2016, pp. 473–478 (2016)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. **11**, 10–18 (2009)
12. Järvelin, K., Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents. In: SIGIR 2000, pp. 41–48 (2000)

[6] https://www.openml.org/d/163.
[7] http://www.openml.org.

13. Kalousis, A.: Algorithm selection via meta-learning. Ph.D. Dissertation (2002)
14. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: interactive visual specification of data transformation scripts. In: CHI 2011, pp. 3363–3372 (2011)
15. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS 2002, pp. 233–246 (2002)
16. Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J. (eds.): Machine Learning: Neural and Statistical Classification. Ellis Horwood, Chichester (1994)
17. Munson, M.A.: A study on the importance of and time spent on different modeling steps. ACM SIGKDD Explor. Newsl. **13**(2), 65–71 (2012)
18. Nguyen, P., Hilario, M., Kalousis, A.: Using meta-mining to support data mining workflow planning and optimization. J. Artif. Intell. Res. **51**, 605–644 (2014)

Systematic Support for Full Knowledge Management Lifecycle by Advanced Semantic Annotation Across Information System Boundaries

Vishwajeet Pattanaik[1(✉)], Alex Norta[1], Michael Felderer[2], and Dirk Draheim[1]

[1] Tallinn University of Technology, Tallinn, Estonia
vpattanaik@gmail.com, alex.norta.phd@ieee.org, draheim@acm.org
[2] University of Innsbruck, Innsbruck, Austria
michael.felderer@uibk.ac.at

Abstract. In today's organizations, there exist a variety of information-system paradigms to support the diverse organizational needs that reside on different levels of operations, tactics and strategies on the one hand, and are subject to different work styles on the other hand. This makes it challenging to design knowledge management systems that can be integrated into heterogeneous information-system applications. In this paper, we present Tippanee that allows users to create and manage semantic annotations over dynamically generated contents across the boundaries of arbitrary information-system applications. We further show, how these advanced semantic annotation capabilities can be systematically exploited in the full knowledge-management lifecycle. We explain, how the essentially important transformation of tacit-explicit-tacit knowledge corresponds to the reorganization of semantic schemata by a mature editorial process.

Keywords: Enterprise resource planning · Knowledge management
Metadata management solutions · Semantic annotations
Social software

1 Introduction

Knowledge management in organizations has been a critical subject of research over decades. Organizations must continuously come up with innovative products and solutions, while seeking and creating new knowledge, and embracing wisdom from society [1–4]. Innovations are becoming more socially dynamic, and rely on simultaneous creation and exploitation of knowledge [5].

While, the massive growth of social media platforms has made it easier to retrieve knowledge from society using crowdsourcing practices [6,7] such as crowd-voting, crowd-funding and microwork. However, designing knowledge

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 66–73, 2018.
https://doi.org/10.1007/978-3-319-92901-9_7

management systems and integrating them into increasing number of information systems applications is becoming a challenging task. On one hand, enterprises are embracing Open Innovation practices [8], while on the other hand knowledge sharing among employees within these enterprises isn't necessarily flourishing [9].

This is because organizations rely on a variety of Information System (IS) applications to support their business activities. While some organizations design their systems and tools themselves, others acquire them from third-party experts. The latter are often independent legacy systems and packages which generally result in heterogeneous software ecosystems. In most cases, large enterprises that build their information systems in-house are able to develop, integrate and maintain their Knowledge Management Systems (KMSs), whereas the same is often impossible for small to medium enterprises (SMEs) as they lack the resources required to perform core KMS activities [10]. Moreover, KMSs are generally designed for a predetermined workflow and lack features to support informal person-to-person communication. Due to which, employees often resort to asking colleagues or improvising in the absence of guidance, habitually missing best practices and repeating mistakes [11]. Employees also might not be prepared to share information in order to protect their jobs or, they might be too busy and may choose not to funnel information into such systems [12].

Considering the aforementioned issues, we derive the following challenges: (1) How to design an independent-interoperable knowledge management tool for organizations with heterogeneous software ecosystems? (2) How to enable knowledge management in a social setting, while using organizational information systems? (3) Can describing things on-the-fly motivate employees to share their knowledge? If so, how? (4) What kind of moderation process would be required to consolidate the knowledge shared by employees?

The Tippanee platform intends to resolve the above mentioned challenges by promoting systematic knowledge creation and management independent of the underlying information systems. The platform is founded on the notion that organizations are social entities and in order to become knowledge creating companies, organizations must encourage knowledge creating and transforming activities while allowing social interactions among employees [1,11,13,14]. Tippanee[1,2] is designed as a lightweight, user-friendly Google Chrome browser extension that allows users to highlight, add, link and share annotations hooked onto elements and processes within HTML documents. Its novel and robust anchoring algorithm detects and reattaches anchors on arbitrarily generated web pages. The platform's semantic annotation features are inspired by the SECI model (socialization, externalization, combination, internalization) of knowledge dimensions [1], therefore it encourages knowledge creation, conversion and transfer. The system enables users to share their tacit knowledge by converting it into semantic descriptions attached to annotated content. Furthermore, users are allowed to create their own semantic vocabularies, making it easier for them to

[1] http://tinyurl.com/tippanee.
[2] https://github.com/vpattanaik/Tippanee.

express their knowledge. Lastly, the system utilizes a moderation process that is somewhat social and includes not just knowledge experts but also the user pool.

The remainder of this paper is structured as follows. Section 2 gives an overview of the related work. Section 3 details the Tippanee platform and its feature from different user prospectives. Finally, Sect. 4 presents a critical discussion about the current stage of our research and concludes the paper.

2 Related Work

2.1 Social Weaver

The core intention of the Social Weaver platform is to enrich existing enterprise applications by weaving them together with end-user applications. The platform enables end users to weave snippets of social software features such as bookmarks, comments and wikis onto the workflow of enterprise applications [14,15]. Unfortunately, since Social Weaver's anchoring algorithm is based on string search, it cannot handle changes in web content. Inspired by Social Weaver, the Tippanee platform extends these social features to enterprise applications while being able to handle content changes, thanks to its novel anchoring approach.

2.2 Fuzzy Anchoring

Used by the annotation platforms Hypothesis[3] and Genius[4], the Fuzzy Anchoring approach is a combination of fuzzy string matching and XML Path (XPath) matching. The approach provides a novel solution for attaching annotations on HTML documents but focuses more on string matching than on structure matching. The fuzzy string matching logic used in the algorithm is a combination of search and compare[5]; and although the algorithm works efficiently on text contents with minor changes, however dramatic changes in content or webpage structure renders the approach useless.

2.3 RDF and Schema.org

Resource Description Framework (RDF) is a data model for representing information about things on the Web. Originally designed as a data model for metadata, the framework allows developers to add metadata about web content. It was designed as a tool for knowledge representation, to assist machine learning algorithm such as search engines and help them better understand, link and present web content[6]. Derived from RDF Schema, Schema.org creates, maintains and promotes a common set of schemas for structured data markup. The

[3] https://web.hypothes.is/.
[4] https://genius.com/web-annotator.
[5] https://web.hypothes.is/blog/fuzzy-anchoring/.
[6] https://www.w3.org/TR/rdf11-primer/.

shared vocabulary provided by Schema.org are developed by an open community process. This shared vocabulary makes it easier for webmasters and developers to add descriptions to web resources[7].

3 Tippanee

Tippanee with its semantic annotation capabilities and design features is meant to encourage employees to create, transform and share knowledge within the organization; following the principles of Knowledge Spiral proposed by Nonaka [1]. The platform supports knowledge creation and facilitates social interactions by means of a user-friendly interface, combined with a robust anchoring algorithm and semantic capabilities. We describe of these features in detail in the following sections.

3.1 User Interface

The Tippanee platform is realized as a Google Chrome browser extension which is designed to be independent i.e., the extension can create and reattach annotations without communicating with the server. Similar to Hypothesis[8], the Tippanee platform functions on a superficial layer above the web page allowing it to be interoperable. When the user opens a webpage on the browser, the extension injects Tippanee's dashboard and its supporting scripts into the HTML Document. The user interface allows users to select web contents even at the sentence level. Annotation are displayed on the right side of the screen, within Tippanee's dashboard. Users can add multiple notes to the annotated text, they can link annotations from different webpages, reconstruct orphaned anchors and add semantic descriptions to annotations. Figure 1, illustrates a snapshot of Tippanee's dashboard.

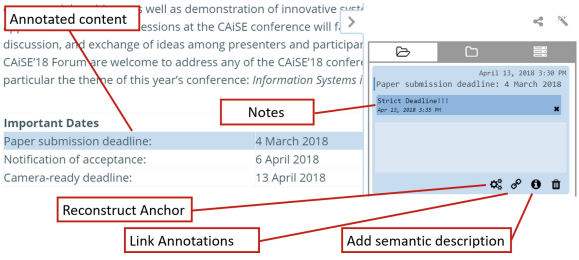


Fig. 1. A section of Tippanee's dashboard, with added note and the *'Describe Anchor', 'Link Notes' and 'Reconstruct Anchor'* buttons on the bottom right

[7] http://schema.org/docs/datamodel.html.
[8] https://web.hypothes.is/about/.

3.2 Anchoring Algorithm

To generate stable anchors, Tippanee's anchoring algorithm analyses HTML Document Object Model (DOM) elements using a predefined set of attributes. When reattaching annotations, the algorithm uses an edit distance approach [16] based on attribute matches and mismatches. This tree matching approach allows for robust anchoring of annotations even on arbitrarily generated webpages. Furthermore, stored anchor information enables preservation and reconstruction of annotations even when they are orphaned.

3.3 Semantics

Tippanee's semantic description feature allows users to describe annotations using Schema.org's metadata data model. The feature is designed keeping two different categories of users in mind. The first category of user is an *employee*. An employee can add simple textual notes and semantic descriptions to annotated content using predefined semantic types and properties. An employee is also allowed to create new semantic types/properties and then describe annotations using the same. The second category of user is the *organization's knowledge expert* hereby referred as the *moderator*. The moderator primarily maintains the knowledge created within the organization. Moderators can use Tippanee as a general employee, however they can also review old-new semantic types and properties. They are allowed to standardize new semantic types and properties based on the ones created by employees or, on the basis of the organizations requirements. Furthermore, moderators are allowed to add, collaborate and restructure the semantic types/properties based on other employees suggestions or the organizations demands.

Features for Employees. The Tippanee platform facilitates employees with two modes: *Plain Annotation Mode* and *Descriptive Mode*. The *Plain Annotation Mode* allows employees to add simple textual notes to annotations. This allows users to highlight and share sections of interests within the Information System. The mode can also be utilized by managers and supervisors to share important announcements and add descriptions to process that might be less understood by users. The feature supports Tippanee's motivation of allowing social interactions within Information Systems, without the need of switching between applications or services. For instance, if a manager would like to inform all his employees of a new feature or a change in the Information System, the manager would usually send an email to the employees informing them about the same. However when using Tippanee, the manager could simply add an annotation to a content on the Information System's homepage and make the annotation viewable to all employees who access the system. This could allows employees to have discussions or ask questions over the same annotation. Since employees can reply to annotations on the fly, the annotations would promote social interactivity. Also, new employees who join the organization at a later period could simply read the annotations and the replies and could update themselves without actually asking for help.

The *Descriptive Mode* is an extension to the *Plain Annotation Mode* and allows users to describe their annotations using semantic descriptions. The Tippanee platform utilizes Schema.org' metadata data model and therefore describes semantic descriptions as 'types' and 'properties'. 'Types' generally represent an item/thing and can have several properties that describe them. The 'properties' are keys to values but can also be described using a 'type'. This means that values of a property could also be complex values which are bundles of other 'properties'.

For instance, lets consider the following description of an organization:
organization: {companyname: *text*, sameas: *text*, ownedby: {person: *firstname: text*, *lastname: text*}}.

The description starts with the type 'organization', which has 'companyname', 'sameas', 'ownedby' as its properties. The properties 'companyname' and 'sameas' are described as simple key-value pairs where the values are of text type. However, the 'ownedby' property is further described using the type 'person', which is then described through its properties 'firstname' and 'lastname'. These are further described using key-value pairs with text type values.

The *Descriptive Mode* is further distinguished into two modes: *Strict Descriptive Mode* and *Flexible Mode*. In the *Strict Descriptive Mode*, the user can only use already available types and properties, whereas the *Flexible Mode* allows customizations to previously available semantic vocabulary, further enhancing the user's knowledge creation capabilities. While creating new vocabularies the user can either decide to do it systematically or as ad-hoc. Adding *Systematic* vocabulary means that the user has to clearly define the added types and properties. The user can either add properties to predefined types and then add values to the new properties; or introduce new types, define their properties and then add values. This way of adding vocabulary is meant for users who understand the organizations standard semantic vocabulary but can't find the appropriate types/properties required to describe specific processes. Adding *ad-hoc* vocabulary means that the user can either simply add key-value pairs independent of the available types and properties, or can add just a key or just a value. This way of adding vocabulary is meant for users who just want to describe their annotations semantically but are not interested in providing or at the time can't think of an appropriate description. Adding ad-hoc vocabulary might also be useful in situations where a user might want to make a quick suggestion, but does not want to waste time looking through the available vocabulary.

Features for Knowledge Experts. The knowledge expert or moderator is a person who understands the semantic vocabulary of the organization and is responsible for updating and maintaining the vocabulary available within the Tippanee platform. It is the expert's responsibility to study the new vocabularies suggested by other employees and organize them according to the organizations requirements. These semantic vocabularies may change from time-to-time depending upon the organizations goals, agendas and processes. However, since the consolidation process suggested by Tippanee is based on social processes, the

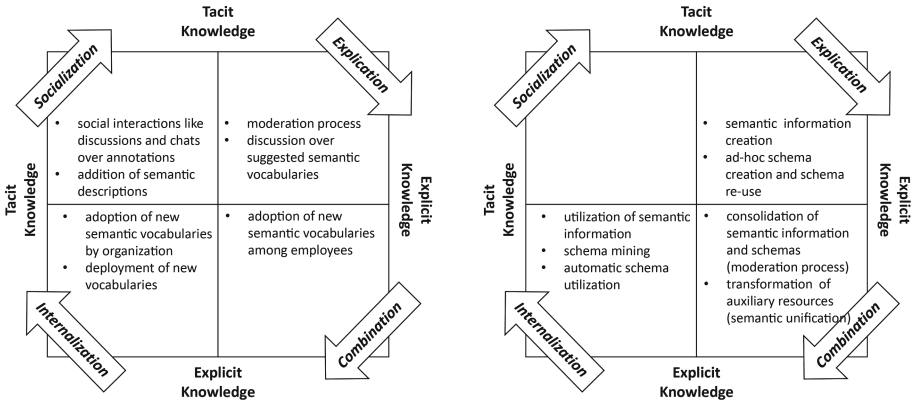


Fig. 2. Lower level *(left)* and higher level *(right)* annotation activities in Tippanee in coordination with the SECI model.

platform tries to include the community in improving the semantic vocabulary; the moderator can look at suggestions provided employees in order to improve the already available semantic vocabulary.

It is vital and interesting to understand that Tippanee's semantic description feature utilizes Nonaka's SECI model at two different levels. At the lower level, the *Plain Annotation Mode* and *Strict Descriptive Mode* allow users to share knowledge in a more structured way, whereas at the higher level the*Flexible Mode* allows sharing of unstructured/semi-structured knowledge; as illustrated in Fig. 2.

4 Conclusion

The Tippanee platform is designed with the intention to introduce Nonaka's concept of a Knowledge Creating Company into organizations' heterogeneous software ecosystems. Currently, the platform is in its early stage of development. The current version of Tippanee available on Chrome Web Store, illustrates our novel-robust-interoperable anchoring approach. The extension allows individual users to add, link, visualize and semantically describe annotations. However, the social features and moderation process are still in development.

On completion, Tippanee would allow users to create, share and transform their tacit knowledge into explicit knowledge by the means of semantics. The platform's social environment would encourage users to participate in knowledge sharing and would allow them to semantically annotate and share process descriptions, further encouraging them to create and share knowledge, not just with work-groups but within the organizations' as whole. Tippanee would further enhance the social aspects of knowledge creation through its moderation process. It would be interesting to see how employees from different departments would

participate as a larger community. Also this would encourage interdepartmental knowledge sharing, enhancing both the employees' and the organizations' knowledge.

References

1. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, Oxford (1995). Everyman's library
2. Nonaka, I., Toyama, R., Hirata, T.: Managing Flow: A Process Theory of the Knowledge-Based Firm. Palgrave Macmillan, Basingstoke (2008)
3. von Krogh, G., Nonaka, I., Rechsteiner, L.: Leadership in organizational knowledge creation: a review and framework. J. Manag. Stud. **49**(1), 240–277 (2012)
4. Leonardi, P.M.: The social media revolution: sharing and learning in the age of leaky knowledge. Inf. Organ. **27**(1), 47–59 (2017)
5. Nonaka, I., Kodama, M., Hirose, A., Kohlbacher, F.: Dynamic fractal organizations for promoting knowledge-based transformation a new paradigm for organizational theory. Eur. Manag. J. **32**(1), 137–146 (2014)
6. Oliveira, F., Ramos, I.: Crowdsourcing: a tool for organizational knowledge creation. In: Twenty Second European Conference on Information Systems (2014)
7. Kucherbaev, P., Daniel, F., Tranquillini, S., Marchese, M.: Crowdsourcing processes: a survey of approaches and opportunities. IEEE Internet Comput. **20**(2), 50–56 (2016)
8. West, J., Salter, A., Vanhaverbeke, W., Chesbrough, H.: Open innovation: the next decade. Res. Policy **43**(5), 805–811 (2014). Open innovation: new insights and evidence
9. Wang, S., Noe, R.A., Wang, Z.M.: Motivating knowledge sharing in knowledge management systems: a quasifield experiment. J. Manag. **40**(4), 978–1009 (2014)
10. Nunes, M.B., Annansingh, F., Eaglestone, B., Wakefield, R.: Knowledge management issues in knowledge-intensive smes. J. Doc. **62**(1), 101–119 (2006)
11. Hemsley, J., Mason, R.M.: Knowledge and knowledge management in the social media age. J. Organ. Comput. Electron. Commer. **23**(1–2), 138–167 (2013)
12. Kaplan, A.M., Haenlein, M.: Users of the world, unite! The challenges and opportunities of social media. Bus. Horiz. **53**(1), 59–68 (2010)
13. Wagner, D., Vollmar, G., Wagner, H.T.: The impact of information technology on knowledge creation: an affordance approach to social media. J. Enterp. Inf. Manag. **27**(1), 31–44 (2014)
14. Draheim, D., Felderer, M., Pekar, V.: Weaving social software features into enterprise resource planning systems. In: Piazolo, F., Felderer, M. (eds.) Novel Methods and Technologies for Enterprise Information Systems. LNISO, vol. 8, pp. 223–237. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07055-1_18
15. Draheim, D.: The present and future of large-scale systems modeling and engineering. In: Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E. (eds.) FDSE 2016. LNCS, vol. 10018, pp. 355–370. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48057-2_25
16. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. **10**, 707 (1966)

Evaluation of Microservice Architectures: A Metric and Tool-Based Approach

Thomas Engel[1,2], Melanie Langermeier[2(✉)], Bernhard Bauer[2], and Alexander Hofmann[1]

[1] MaibornWolff GmbH, Munich, Germany
[2] University of Augsburg, Augsburg, Germany
melanie.langermeier@informatik.uni-augsburg.de

Abstract. Microservices are an architectural style that decomposes the functionality of an application system into several small functional units. The services are implemented and managed independently from each other. Breaking up monolithic structures into a microservice architecture increases the number of single components massively. Thus, effective management of the dependencies between them is required. This task can be supported with the creation and evaluation of architectural models. In this work, we propose an evaluation approach for microservice architectures based on identified architecture principles from research and practice like a small size of the services, a domain-driven design or loose coupling. Based on a study showing the challenges in current microservice architectures, we derived principles and metrics for the evaluation of the architectural design. The required architecture data is captured with a reverse engineering approach from traces of communication data. The developed tool is finally evaluated within a case study.

Keywords: Architecture evaluation · Microservices · Metrics
Microservice principles

1 Introduction

Microservices are an emerging style for designing software architectures to overcome current issues with monoliths like the difficulties of maintenance and system evolvement, but also their limited scalability. The style is characterized by building an application through the composition of independent functional units, running its own process, and communicating through message passing [1]. The microservice architectural style enables the creation of scalable, reliable and agile software systems [2]. With an automatic deployment, they enable shorter product development cycles and serve the need for more flexible software systems.

Microservice systems can be composed of hundreds or even thousands of services [3]. Complexity increases also due to the nature of a distributed system and the typically high release frequency [4]. Managing and monitoring those systems is essential [2–4]. This includes the detection of failures and anomalies

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 74–89, 2018.
https://doi.org/10.1007/978-3-319-92901-9_8

at runtime but also ensuring a suitable architecture design for example regarding consistency and fault tolerance [2].

Organizations employing microservice architectures require a "systematic understanding of maintainability as well as metrics to automatically quantify its degrees" [4]. Analyzing the performance of the system and understanding failures that occur during runtime is essential to understand the system as a whole [5]. Existing measurement approaches for service- or object-oriented systems are difficult to apply within microservice architectures [4]. Since microservice systems can contain up to thousands of single services, the architectural models get very large and complex. If they are manually created, typical problems in practice are inconsistent and incomplete models. Often information is missing or only a part of the architecture is represented. The up-to-dateness of the data is also questionable in many cases. Recovering the architecture from source code or other artifacts is seen as one option to improve the management of microservice systems [3]. However, source code is not always available or a great diversity of different programming languages hinders its analysis.

In the following we present a method and tool for the evaluation of microservice architectures based on a reconstructed architectural model. In a first step, we analyzed current literature (Sect. 2) and five microservice projects (Sect. 3) for microservice architecture principles. From these principles we derived the metrics that are used to evaluate the architecture and to identify hot spots (Sect. 4.1). Hot spots indicate parts of the architecture, where problems may occur and thus require a detailed consideration. The required architectural model is reconstructed based on communication data to ensure its correctness and completeness (Sect. 4.2). Based on these concepts we implemented the Microservice Architecture Analysis Tool MAAT (Sect. 5) and integrated it into an existing project for evaluation purposes (Sect. 6).

2 Foundations

2.1 Characteristics of Microservices Systems

A microservice system is a decentral system that is composed of several small services that are independent from each other. The communication takes place via light-weight mechanisms i.e. messaging. Instead of a central orchestration of the services, a decentral management like choreography is applied. Core principles of a microservice architecture are loose coupling and high cohesion [1,6–8]. Microservice architectures follow a domain-driven design, where each microservice is responsible for one bounded context [2,8–10]. That means, that a microservice provides only a limited amount of functionality serving specific business capabilities. Microservices are developed independently from each other i.e. they are independent regarding the utilized technologies and programming languages, the deployment and also the development team [2,10–12]. A team is responsible for the full lifecycle of a microservice, from development to its operation [1]. The build and release process is automated and enables continuous delivery [1,8,11]. According to [5] scalability, independence, maintainability

(including changeability), deployment, health management and modularity (i.e. single responsibility) are the most mentioned attributes of microservice systems in literature.

During implementation of a microservice system, two major issues have to be considered. The first one addresses challenges due to the nature of distributed systems and the second one the context definition of a microservice. Finding the right granularity of microservices is essential, since badly designed microservices increase the communication complexity of the system [2,10,11]. They also reduce the extensibility of the system and impede later refactoring and the overall management.

Since communication is essential in such a system, network communication, performance and complexity are important issues [1,5,11,12]. The system must be able to deal with latency and provides mechanisms for debugging, monitoring, auditing and forensic analysis. The architectural design has to support the openness of the system and has to deal with the heterogeneity [1,13]. Additionally, fault handling and fault tolerance [5,13] as well as security issues [1] must be taken into account.

2.2 Service Metrics for Architecture Evaluation

Scenario-based or metric-based evaluation is required to assess the quality of attributes of a software system and provide a solid foundation for planning activities [14]. There exists a large number of metrics for measuring and evaluating services and service-based systems. [14] provides an overview of the metrics in current literature with focus on maintainability and analyzes their applicability for microservice systems. With exception of centralization metrics the majority of the identified metrics are also applicable for microservice systems. But specific evaluation approaches for microservice systems are rare and especially practicable and automatic evaluation methods are missing [4].

In [4] a maintainability model for service-oriented systems and also microservice systems is presented. For the identified quality attributes the authors determine applicable metrics to measure the system architecture. For example, the coupling degree of a service can be measured considering the number of consumed services, the number of consumers and the number of pairwise dependencies in the system. Metrics for measuring cohesion are the diversity degree of parameter types at the interfaces. A highly cohesive service can also be identified considering the distribution of using services to the provided endpoints. Granularity is for example measured with the number of exposed interface operations. Further quality attributes are complexity and code maturation. A weakness of [4] is the missing tool support and integration into the development process.

In [15] constraints and metrics are presented to automatically evaluate microservice decomposition architectures. The authors define metrics to assess pattern conformance regarding decomposition. They focus on two aspects, the independence of the microservice and the existence of shared dependencies or components. The metrics and constraints rely on information about the connector type, i.e. whether it is an ‘in-memory’ connector or a loosely coupled

connector. For example, to quantify the independence degree the ratio of the number of independent clusters to the number of all non-external components is used. Ideally the cluster has the size of 1 and thus the resulting ratio is 1. Components are aggregated to a cluster if they have an in memory connection.

3 Challenges Within Microservice Architectures in Practice

Within an exploratory study we analyzed the challenges in five different microservice projects. Table 1 presents the projects with the their type, duration, size (in number of microservices, MS) and the utilized technologies. We conducted in-depth interviews with selected experts from the projects.

Table 1. Project details

Project	Project type	Duration	Size	Technologies
Pr1	Big data application	2.5 years	20–30 MS	Cassandra, Elastic, HiveMQ, Payara, PostgreSQL, Spark
Pr2	Cloud platform	1 year	3 MS	Akka, PostgreSQL, Spring Boot
Pr3	Cloud platform, Smart Home	6 months	4 MS	Docker, Kubernetes, RabbitMQ, Vert.x
Pr4	ETL project	6 weeks	2 MS	Docker, Kafka, Kubernetes, Openshift
Pr5	Information system	2 years	ca. 50 MS	AWS, consul, Docker, FeignClient, MongoDB, RabbitMQ, Spring Boot

An interview constitutes of four parts: In the introduction the purpose of this study and the next steps were presented. Afterwards the interview partner was asked to describe the project including her own role in the project. The first main part of the interview was about identifying challenges and problems within the project. This included also the reason for choosing a microservice architecture. Finally an open part was left to discuss project-specific topics in detail, that were not covered so far. Resulting topics were e.g. the size of microservices, rules for the decomposition into small services, use cases covering multiple microservices and a refactoring affecting more than one microservice. For some projects asynchronous communication, data consistency, deployment and testing were addressed as well.

Especially in larger projects, context definition of microservices was mentioned as a major challenge (Pr1, Pr5). Questions in this context were ‘What is

a microservice?', 'What size does it have?' and 'Which tasks does the microservice implement?'. With these questions as baseline the problems arising through a unsuitable service decomposition were elaborated. Discussed consequences are an increasing communication complexity which leads to performance weaknesses and the difficult integration of new features that lead to an increasing time-to-market. An unsuitable breakdown of microservices tends to have more dependencies and thus the developer has to understand greater parts of a system for the implementation of a new feature.

Another challenge is refactoring, that affect more than one microservice (Pr1, Pr5). In contrast to monolithic systems, there is no tool support to perform effective refactorings of microservice systems. Additionally the complexity increases through the utilization of JSON objects or similar formats at the interfaces.

Closely related to that aspect is the issue of non-local changes (Pr1, Pr3). Experiences in the projects have shown that changes made to one microservice may not be limited to this microservice. For example changing interface parameters has effects on all microservices using that interface. Another reason why changes in one microservice may lead to changes on other services is because of changes in the utilized technologies. Experience in the projects have shown that the adherence of homogeneity is advised for better maintainability, robustness and understandability of the overall system. Even if the independence of technology and programming languages is possible and a huge advantage, system-wide technology decisions should be made, and only if necessary deviations should be made. This is why changes affecting the technology of one microservice may also lead to changes in other microservices.

A challenge that addresses the design of microservice systems are cyclic dependencies (Pr1). These dependencies cannot be identified automatically and cause severe problems when deploying new versions of a service within a cycle. To prevent a system failure all services the compatibility of the new version to all services in the cycle has to be ensured.

Keeping an overview of the system is a big challenge in the projects (Pr1, Pr4, Pr5). Especially large microservice systems are developed using agile development methods within different autonomous teams. Each team is responsible for its own microservices, carries out refactorings and implements new functionality by its own. Experiences in the projects show that it is hard to not lose track of the system. The complexity of this task increases, since the dataflow and service dependencies cannot be statically analyzed like in monolithic systems.

Beside the above challenges, monitoring and logging (Pr1, Pr4), data consistency (Pr4) and testing (Pr5) were also mentioned in the interviews. Monitoring and logging address the challenge of finding, analyzing and solving failures. Testing is aggravated by the nature of a distributed system and the heavy usage of asynchronous communication causing large problems concerning data consistency.

4 Evaluating Microservice Architectures

To address the identified challenges in current microservice projects we propose a tool-supported evaluation and optimization approach. The proposed method contains four steps (Fig. 1). Since we do not make the assumption of having an up-to-date architectural model, the first step is retrieving the required architectural data. Upon this information an architectural model is built, representing the microservice system with the communication dependencies. The model is used to evaluate the architecture design based on identified principles with corresponding metrics. Finally the results are visualized using a dependency graph and coloring.

Fig. 1. Process for the evaluation of microservice architectures

In the following the retrieved principles with their metrics are presented (Sect. 4.1) as well as the data retrieval approach (Sect. 4.2). The visualization and the architectural model are shown exemplary for our tool in Sect. 5.

4.1 Evaluation Criteria: Principles and Metrics

Based on the literature (Sect. 2) and the results of the structured interviews (Sect. 3) we identified 10 principles for the design of microservice architectures. Table 2 provides an overview of them with their references to literature and the related projects.

To evaluate the principles, we derive metrics using the Goal Question Metric (GQM) approach [18]. To ensure a tool-based evaluation the metrics should be evaluable with the automatically gathered information in the first two method steps. For the metric definition we considered the proposed metrics by [4] to measure quality attributes of a service-oriented system like coupling, cohesion and granularity. The metrics proposed in [15] were not practicable for our approach, since the required information cannot automatically gathered from communication logs. Especially in-memory dependencies are not graspable.

The overall goal is the adherence of the principles. A possible question in this context is for example '*Are the microservices of a small size?*' (P1). The derived metric for the small size principle counts the number of synchronous and asynchronous interfaces of a service (M1). A (a)synchronous interface indicates a provided functionality of a service via an endpoint, that can be consumed synchronously (respectively asynchronously). The number of interfaces is an indicator for the amount of implemented functionality and can thus be used to measure the size of a microservice. Due to the relation to the implemented functionality

Table 2. Principles for microservice architectures

Nr	Principle	Mentions	Corresponding metric
P1	Small size of microservice	[1,6–8], Pr2, Pr5	M1: #(A)synchronous interfaces
P2	One microservice - one task (SoC, SRP)	[1,7,16], Pr1, Pr2	M1: #(A)synchronous interfaces M2: Distribution of synchronous calls
P3	Scalability	[5,16], Pr1, Pr5	M2: Distribution of synchronous calls
P4	Loose coupling and high cohesion	[1,6], Pr1, Pr3	M3: #(A)synchronous dependencies
P5	Domain-driven design	[9,10], Pr1, Pr2, Pr3, Pr5	M3: #(A)synchronous dependencies M4: Longest synchronous call trace M5: Average size of asynchronous messages
P6	Manageability/Clarity of the system (system design)	[5,10,11], Pr1	M3: #(A)synchronous dependencies
P7	Low network complexity	[1,10], Pr2	M3: #(A)synchronous dependencies M6: #Synchronous cycles
P8	Independence (e.g. technology, lifecycles, deployment)	[1,2,5,8,10–12], Pr1	
P9	No cyclic dependencies	[17], Pr1	M6: #Synchronous cycles
P10	Performance	[5,11,12], Pr1	M4: Longest synchronous call trace M5: Average size of asynchronous messages

the metric can also be used for evaluating P2 (one task per microservice). Additionally, the distribution of the synchronous calls (M2) can be used to determine services that fulfill several tasks. Therefore the quotient of the minimum and maximum amount of synchronous calls at the interfaces of a service is considered. Having a small quotient, the number of interface calls varies widely and the microservice may implement more than one task. A high diversity within the number of interface calls hampers scalability (P3). Executing several instances of a service to deal with the high number of incoming request leads also to a duplication of the functionality behind the less requested interfaces.

To evaluate the principle loose coupling and high cohesion (P4) the number of synchronous and asynchronous dependencies (M3) can be used. This metric is calculated at service level and denotes the number of service calls, either via synchronous or asynchronous communication dependencies. This metric is also used to assess the principle of domain-driven design (P5). If a system is built around business capabilities, the services own the majority of the required

data by themself. In contrast a system built around business entities contains more dependencies since more data has to be exchanged. A further indicator that speaks against a domain-driven design is a long synchronous call trace. M4 provides the longest synchronous call trace identified in the microservice system.

The principles P6 and P7, small network complexity and manageability of the system, can be evaluated using the number of (a)synchronous dependencies (M3). A small amount of dependencies indicates a low complexity of the network, while it also supports the clarity of the whole system. The network complexity increases, if there exist cyclic communication dependencies. M6 provides the number of strongly connected components in the microservice system in order to refine the evaluation of P7. Thereby only the synchronous dependencies are considered, since they cause blocking calls. This metrics is strongly connected to the principle P9 which says that a microservice systems should not contain cyclic dependencies.

To provide indicators for the system performance (P10) the longest synchronous call trace (M4) and the average size of asynchronous messages (M5) are proposed. Large messages may cause performance bottlenecks and the problem of network latency increases with a rising number of services within one call trace. Both factors have negative impact on the overall performance of a microservice system. Finally, P8 describes the independence of the microservice in different aspects like technology, lifecycles, deployment but also the development team. With the concept of utilizing communication and trace data we are currently not able to provide a meaningful metric. Integrating static data e.g. from source code repositories in future work can address this issue.

4.2 Data Retrieval

For the application of the metrics the architectural model must contain the following information: The microservices with their names and the provided interfaces. The number, size and source of incoming requests at an interface. A differentiation between synchronous and asynchronous has to be available for the communication dependencies. For synchronous requests the respective trace information is required in order to interpret the dependencies correctly. Finally it is recommended to aggregate the microservices within logical groups (clusters). Clustering enables an aggregation of characteristics to higher abstraction levels and thus supports the understandability of the architectural model.

To be able to perform the evaluation on an up-to-date architectural model, we decided to employ a reverse engineering approach. These approaches provide techniques to derive a more abstract representation of the system of interest [3]. [3] provide a two-step extraction method with a final user refinement of the architectural model. They include a static analysis of the source code as well as a dynamic analysis of the communication logs. The focus for our evaluation are the communication dependencies between the services. We decided to derive the required information from the communication data. Later extensions can include further information from the source code repository. With this approach, we are able to create an architectural model representing the actual behavior of

the system. We do not have to estimate performance measurements like the number of messages or the message size. Furthermore, we do not have to deal with the heterogeneity of the technologies and programming languages regarding communication specification. Another option would be the utilization of manual created models. In this case the number and size of message has to be estimated and additionally, these models are of often not up to date.

When implementing a dynamic approach for data retrieval, it is important to set an adequate time span for data retrieval, since only communicating services are captured. Additionally, the data retrieval procedure must be performed in a way, that it has no significant impact on the productive system. Especially the performance must be ensured, when integrating such a logging approach. Due to the independence and heterogeneity of microservices, the data retrieval must also be independent from the used technologies and programming languages. And finally, the necessary extensions have to be performed without changing the source code of the microservice. The last point is important to ensure the practicability and acceptance of the approach.

We retrieve the required data from the communication logs of the microservice system. The communication dependencies are retrieved from logging the synchronous and asynchronous messages. The communication endpoints are identified as interfaces. Sources for the data are application monitoring tools and message brokers for aggregated data or the Open Tracing API to record service calls. The logical aggregation of microservices into clusters in made manually. After establishing the architectural model, a review has to ensure the quality of the model. The recovered architectural model contains only those services, that send or receive a message during the logged time span.

5 Tool for Architecture Evaluation: MAAT

For metric execution we implemented a tool, considering the requirements for data retrieval from the previous section. The Microservice Architecture Analysis Tool (MAAT) enables an automatic evaluation of the architectural design. The tool provides a visualization of the system within an interactive dashboard and applies the metrics described in Sect. 4.1. Additionally, it provides interfaces for the retrieval of communication data and an automatic recovery of the architectural model. The main goals are providing an overview of the system and to support the identification of hot spots in the architecture as well as the analysis of the system. Therefore, it can be used within the development process to ensure the quality of the implemented system but also in software and architecture audits. Figure 2 shows the architecture of MAAT.

Data retrieval is done within the *Data collector* component. This component provides two interfaces, one implementing the Open Tracing Interface to capture synchronous communication data and one for logging the messages from asynchronous communication. The *Data collector* processes this raw information and writes it into the database. Further information about the data retrieval

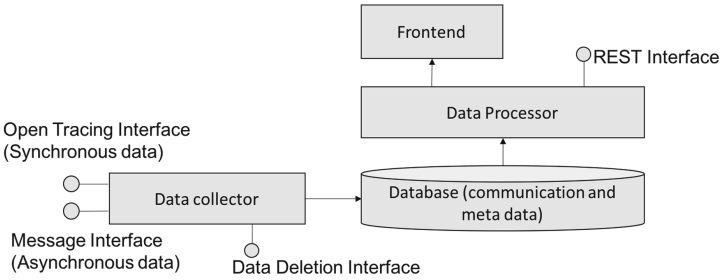


Fig. 2. MAAT overview

at the interfaces are provided in the next two sections. The *Data processor* component utilizes the communication data in the database to generate a dependency graph of the system and to calculate the metrics. The results can be accessed via an implemented *Frontend* but also via a REST interface. The *Frontend* is responsible for the visualization of the dependency graph and the presentation of the metric results. The data and metric visualization is described in Sects. 5.3 and 5.4.

5.1 Synchronous Data Retrieval

For synchronous data retrieval, the Open Tracing standard[1] is used. Open Tracing is a vendor neutral interface for distributed tracing that provides an interface for collecting trace data and is widely used. The reason for this standard is, that there exist many distributed tracing systems collecting more or less the same data but using different APIs. Open Tracing unifies these APIs in order to improve the exchangeability of these distributed tracing systems. Zipkin[2] implements the Open Tracing Interface for all common programming languages like Java, C#, C++, Javascript, Python and Go and even for some technologies like Spring Boot. The *Data collector* of MAAT implements the same interface as Zipkin. Thus, the libraries developed for Zipkin can also be used for MAAT to collect synchronous data.

5.2 Asynchronous Data Retrieval

In contrast to synchronous data retrieval there is no standard available for collecting asynchronous data. Therefore, we had to implement our own interface for asynchronous data retrieval that is independent of any message broker. The *Data collector* of MAAT implements an interface for asynchronous data expecting the microservice name, the message size, an identification of the endpoint e.g. the topic, the name of the message broker, a timestamp and the type indicating if it was a send or received message. The generic interface is implemented

[1] http://opentracing.io/.
[2] http://zipkin.io/.

for RabbitMQ using the Firehose Tracer[3] to get the data. If the Firehose Tracer is activated, all send and received messages are also send to a tracing exchange and enriched with additional meta data. A specific RabbitMQReporter is implemented as intermediate service between the MAAT *Data collector* and the message broker RabbitMQ. The task of the RabbitMQReporter is to connect to the tracing exchange of RabbitMQ and to process all messages. He extracts the required information from the messages, finds the name of the microservice that has send respectively received the message and calls the interface of the MAAT *Data collector* to forward the information.

5.3 Data Visualization

The architecture is visualized as dependency graph. Nodes represent microservices and edges represent (a)synchronous dependencies. A solid edge with arrow at the end indicates a synchronous call and a dashed edge with arrow in the middle an asynchronous message. A microservice cluster is identified with a grey rectangle in the background. Figure 3 shows an example representation of a microservice system.

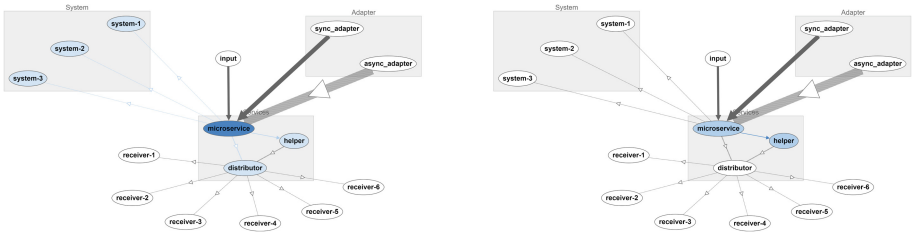


Fig. 3. Architecture visualization and context specific element highlighting

To support navigation and understandability in large microservice systems, a context sensitive neighborhood highlighting is implemented. Selecting a specific microservice, all outgoing dependencies as well as the respective target services will be highlighted. In Fig. 3 on the left the service named microservice is selected. All directly dependent services (outgoing communication edges) are highlighted to. In the right part, the edge is selected and respectively source and target are highlighted. This functionality adapts also to the respective metrics under consideration, i.e. if only synchronous dependencies are considered also only the synchronous neighbors are emphasized.

Additionally, the tool enables the creation of a more abstract view by aggregating the dependencies between clusters. This view provides a high-level overview of the system. If required the user can step into details by dissolving a single aggregated dependency or all aggregated dependencies for a specific cluster.

[3] https://www.rabbitmq.com/firehose.html.

5.4 Metric Evaluation

Despite a textual representation of system and service details the tool visualizes the metric results within the dependency graph. Except M6 each metric refers to one microservice and therefore microservices are marked using the colors green, yellow and red depending on the calculated result. This color coding is intuitive, emphasizes the metric evaluation and hot spots can be quickly identified. Figure 4 shows the evaluation of the metric M3, number of asynchronous dependencies, in a fictional system. Since the service *distributor* has more than 5 asynchronous dependencies, he is colored red and the service *microservice* is colored yellow because he has between 4 and 5 asynchronous dependencies. All other microservices have 3 or less asynchronous dependencies and therefore colored green.

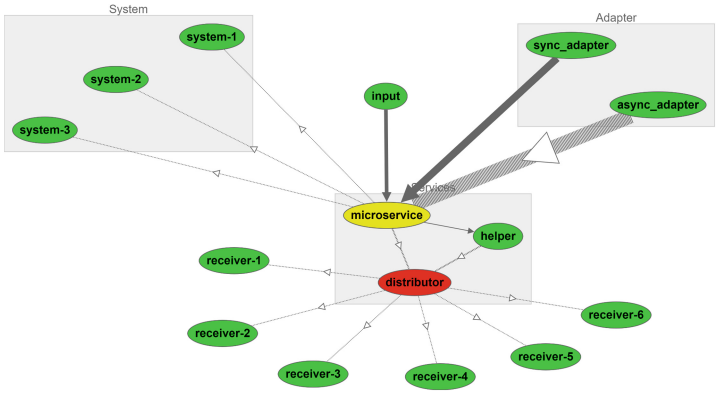


Fig. 4. Visualization of metric results (Color figure online)

Since M6, the number of synchronous cycles, is bound to the whole system the visualization of the metric evaluation differs from the other ones. In this case all microservices and dependencies belonging to a cycle are marked with one color in order to easily spot cycles and distinguish between them.

6 Evaluation and Discussion

For evaluation purposes, we integrated the tool in a microservice project (Pr5) and applied the proposed method. The results are discussed with team members from the project. Additionally, we discuss related work and the strength and weaknesses of the approach.

The application system of the case study consists of about 50 microservices. Utilized technologies in the project are AWS, consul, Docker, FeignClient, MongoDB, RabbitMQ and Spring Boot. The project is under development for two years and is developed by three scrum teams at different locations. To gather the

asynchronous communication data the proposed RabbitMQReporter in Sect. 5 is integrated into the system. Other reporter services are also possible at this point. For synchronous data Spring Cloud Sleuth[4], an implementation of the Open Tracing API, is used. An alternative, independent from Spring Cloud would be brave[5] at this point. In this project, environmental variables are used to set the configuration for different target platforms. Thus, the service responsible for the RabbitMQ configuration was extended as well as the microservice variables for Spring Boot. For the integration of MAAT, no changes to the source code were necessary. Small changes have to be made for each microservice regarding the deployment. Altogether the integration was done in 10 man-days.

After the tool integration, data retrieval was performed during a system test. Thereby 40 microservices were identified, which were clustered in 9 groups. 3.466 synchronous calls could be aggregated to 14 different communication dependencies and another 49.847 asynchronous calls to 57 dependencies. Gathering the data at runtime can have impact on the performance. According to RabbitMQ the overhead for synchronous data retrieval in only additional messages. For synchronous data it is possible to specify the percentage of traced calls in order to address this issue.

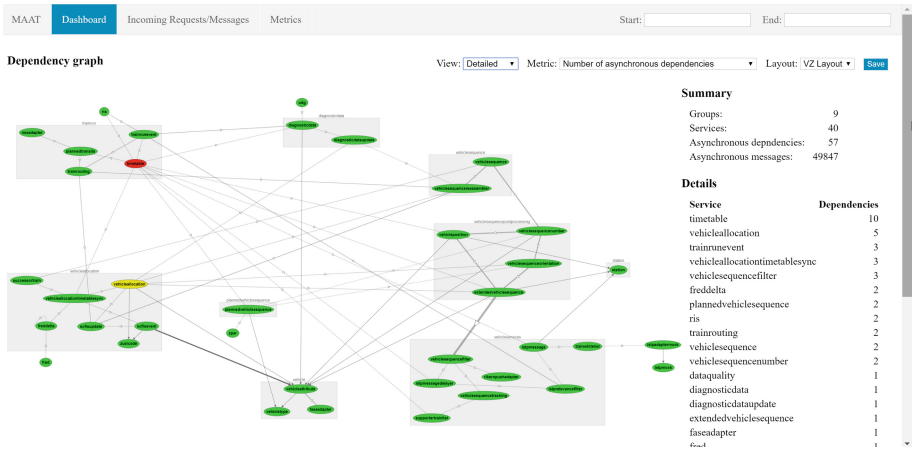


Fig. 5. Application of the evaluation tool in the case study (Color figure online)

The metrics were executed using the retrieved data and discussed with an architect of the development team. In Fig. 5 the evaluation of the metric 'Number of asynchronous dependencies' is presented. According to the result, the services are colored with green, yellow and red. The service marked red in the figure does not fulfill the principle of loosely coupled services. The high number

[4] https://cloud.spring.io/spring-cloud-sleuth/.
[5] https://github.com/openzipkin/brave.

of asynchronous dependencies together with its name (*timetable*) indicates a violation of the principle domain-driven design, where the service is built around an entity. Further implications from the evaluation were a missing data caching at a microservice and a missing service in the overall system test that was not visualized in the architectural model. The overall architecture was evaluated as good by the architects. This was also represented by the metrics. Nevertheless, several discussion points were identified during the evaluation (see points above).

The tool integration was possible without making changes to the source code. After recovering the architectural model from the communication data, a review was required and only minor corrections had to be made. The evaluation results, i.e. the visualization of the architecture and the metric execution, advance the discussion about the architectural design. The discussion outlines issues, that were not in mind of the team before. All over the positive outcome of the metric evaluation confirms the perception of the team, having a quite good architecture.

The single disciplines of architecture recovery and service evaluation are presented in Sect. 2. Approaches that combine both disciple in some way a distributed tracing systems and application performance management (APM) tools. Distributed tracing system, e.g. Zipkin, collect data at runtime in order to identify single requests and transactions and to measure the performance. The focus here is the runtime behavior of the system. MAAT provides a static view on the communication data with additional concepts for handling the models. APM tools are also analysis tools for microservice systems. Their focus is a real-time monitoring to support the operation of those systems. MAAT is primarily used by architects and developers for design and development of those systems.

7 Conclusion

Within an exploratory study of five microservice projects we identified several challenges. Context definition of microservices, keeping an overview of the systems and the effects of refactorings and non-local changes are major issues in the projects. Additional cyclic dependencies, monitoring and logging, consistency and testing are mentioned topics by the interviewed experts. To address these challenges we proposed a tool-supported evaluation method for microservice architectures. Relying on the dynamic communication data we recovered a static dependency model of the current architecture. Additionally we determined principles for the microservice architecture design from literature and the experiences in the considered projects. Using the Goal Question Metric approach we derived automatic executable metrics to evaluate the recovered dependency model. The visualization of the metric results within the implemented tool MAAT provides a quick access to design hot spots of the architecture. The method and the tool are applied within one of the projects. For further evaluation of the quality of the metrics to validate a microservice architecture, their application in further use cases is required. Since we were not able the evaluate the independence of the microservices in the different identified aspects, future work has to elaborate the integration of static information, e.g. from the source

code repository. Future work will include the combination of the dynamically derived data with static data, e.g. source code, as proposed by [3] to improve the significance of the metrics. Additionally minor improvements like further support for service clustering and dynamic coloring of the metrics are planned.

References

1. Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow. In: Mazzara, M., Meyer, B. (eds.) Present and Ulterior Software Engineering, pp. 195–216. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67425-4_12
2. Hasselbring, W., Steinacker, G.: Microservice architectures for scalability, agility and reliability in e-commerce. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 243–246 (2017)
3. Granchelli, G., Cardarelli, M., Francesco, P.D., Malavolta, I., Iovino, L., Salle, A.D.: Towards recovering the software architecture of microservice-based systems. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 46–53 (2017)
4. Bogner, J., Wagner, S., Zimmermann, A.: Towards a practical maintainability quality model for service-and microservice-based systems. In: Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings, ECSA 2017, pp. 195–198. ACM, New York (2017)
5. 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) (2016)
6. Newman, S.: Building Microservices. O'Reilly Media Inc., Newton (2015)
7. Thönes, J.: Microservices. IEEE Softw. **32**(1) (2015)
8. Amundsen, M., McLarty, M., Mitra, R., Nadareishvili, I.: Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media, Inc., Newton (2016)
9. Evans, E.J.: Domain-Driven Design. Addison Wesley, Boston (2003)
10. Fowler, M., Lewis, J.: Microservices - a definition of this new architectural term (2014). https://martinfowler.com/articles/microservices.html. Accessed 29 Sept 2017
11. Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., Steinder, M.: Performance evaluation of microservices architectures using containers. In: 2015 IEEE 14th International Symposium on Network Computing and Applications (NCA), pp. 27–34 (2015)
12. Francesco, P.D., Malavolta, I., Lago, P.: Research on architecting microservices: trends, focus, and potential for industrial adoption. In: 2017 IEEE International Conference on Software Architecture (ICSA), pp. 21–30 (2017)
13. Coulouris, G.F., Dollimore, J., Kindberg, T.: Distributed Systems. Pearson Education Inc., London (2005)
14. Bogner, J., Wagner, S., Zimmermann, A.: Automatically measuring the maintainability of service- and microservice-based systems - a literature review. In: Proceedings of IWSM/Mensura 17, Gothenburg, Sweden (2017)
15. Zdun, U., Navarro, E., Leymann, F.: Ensuring and assessing architecture conformance to microservice decomposition patterns. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 411–429. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_29

16. Killalea, T.: The hidden dividends of microservices. Commun. ACM **59**(8), 42–45 (2016)
17. Sharma, S.: Mastering Microservices with Java. Packt Publishing Ltd., Birmingham (2016)
18. Basili, V.R., Rombach, H.D.: The TAME project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. **14**(6), 758–773 (1988)

KeyPro - A Decision Support System for Discovering Important Business Processes in Information Systems

Christian Fleig(✉), Dominik Augenstein, and Alexander Maedche

Institute of Information Systems and Marketing (IISM),
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
christian.fleig@kit.edu

Abstract. Prioritization is essential for process decision-making. Organizational decision-makers frequently do not have clear perceptions of which processes are of prime and secondary importance. Further, research does not provide a satisfactory definition of process importance and quantifiable proxies. Thus, this paper derives dimensions of process importance including the number of process executions, involved process stakeholders, customer or supplier involvement, and value chain position from literature, and introduces measurable proxies from information systems for each dimension. The artifact "KeyPro" is developed to explore important processes bottom-up from log data in SAP-ERP systems with visualization in interactive dashboards. As process library, we retrieve 220 processes from 52 process experts across four real-world companies. 773 ERP transactions are matched to these processes to measure which processes constitute important processes along the importance dimensions. We implemented KeyPro in three real-world SAP-ERP systems and a focus group of process experts. An evaluation in two case studies reveals significant deviations between perceptions by decision-makers and the results suggested by KeyPro.

Keywords: Business processes · Process importance
Enterprise resource planning systems · Decision support systems
Design science

1 Introduction

Rapidly evolving competitive environments and new business opportunities require organizations to flexibly adapt their organizational design to new conditions [1]. These increasingly dynamic and constantly evolving organizational surroundings require the transformation on different levels of the organization such as strategy, business models, architecture, technology, and business processes in response to internal and external shifts. In particular, organizations transform business processes to restore the fit between the internal and external environment and the organizational design to remain competitive (e.g., [2–4]), or to achieve various benefits in terms of cost reductions, flexibility and agility, sustainability, efficiency, simplification, standardization, security, compliance or increased transparency. However, as a fundamental prerequisite for any

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 90–104, 2018.
https://doi.org/10.1007/978-3-319-92901-9_9

such process transformation endeavor, organizations require a precise understanding of which business processes are of primary importance to the organization for appropriate transformation decision-making. A comprehensive understanding of which business processes actually are of prime importance with a significant impact on the organizational value creation significantly improves decision-making related to process transformation. A clear perception of "key" business processes enables organizations to prioritize transformation projects to primary processes, to improve investment decisions and resource management by allocating limited resources to value-creating processes. For example, before the start of a process mining project, organizations are required to prioritize processes to make decisions concerning which processes should be implemented in process mining. Further, process prioritization allows to focus managerial attention as well as process monitoring activities to key processes.

However, although business processes are of prime significance for the organizational value creation [5], a large number of organizations does not have an exhaustive understanding of how their own business processes behave in reality [6–8], and of which of the business processes can be considered as "important".

Nowadays, companies largely build their operations on information systems [9] and increasingly generate vast amounts of data (e.g., [10]) in business activity. In particular, organizational information systems (IS) such as Enterprise Resource Planning (ERP), Workflow Management (WfM), Customer Relation Management (CRM) or Supply Chain Management (SCM) systems store large amounts of process-related data [6], which might be used to extract process knowledge [11]. For example, SAP-ERP systems in practice store each executed transaction and associated information in large event log tables. These transaction programs reveal information about associated business processes. Transactions can be mapped to business processes to determine quantifiable importance metrics, and then decide whether a business process seems to be of primary or secondary importance for the organization.

To the best of our knowledge, there is no existing work that previously investigated which business processes are most important to organizations and how such processes can be discovered automatically in a bottom-up approach by relying on data from information systems. Therefore, this paper proposes a design science approach for the development of a decision support system (DSS) "KeyPro" as an alternative bottom-up way to discover and to explore important business processes. Thereby, the approach presented in this paper complements traditional top-down approaches relying on tacit managerial and human knowledge and estimations. To support decision-makers in determining "key" organizational processes, we propose to define the construct of "process importance" in a first step, and then to quantify process importance by automatically computing key processes from information systems in a second step. We leverage existing literature and conceptualize "process importance" as a multi-dimensional construct. In sum, we formulate the research question of this paper as: *"How to design a decision support system to automatically discover and explore important business processes from organizational information systems?"*.

We follow a design science approach as suggested by Vaishnavi and Kuechler [12] In a problem awareness phase, we performed a series of workshops at an industry partner site to determine whether decision-makers believe to have an adequate understanding of which of their processes are of primary and secondary importance for

the organization to validate the previously stated hypothesis of biases in the perception of decision-makers about process importance compared to system reality. The workshops were performed in the context of an SAP S/4 HANA migration and process standardization project with corporate-level managers. As a finding, decision-makers stated they frequently need to rely on top-down intangible knowledge about business processes to determine which of the processes is of primary importance to the organization. Furthermore, existing approaches to process prioritization rely on surveys and interview techniques, which are not funded by data or objective facts, and thus potentially biased. In the suggestion phase, we leverage existing literature to gather demands from academics and organizational decision-makers for a problem solution which overcomes these weaknesses of subjective, top-down approaches to the determination of important business processes.

The remainder of this paper is organized as follows. Section 2 describes conceptual foundations on process importance. Section 3 describes meta-requirements and design principles for the DSS derived from workshops with process decision-makers at the industry partner. Section 4 presents the prototype implementation of KeyPro with data from three real-world SAP R/3-ERP systems at the industry partner based on the process importance conceptualization from Sect. 2. Section 5 presents case study evaluation results specifically analyzing the processes of two different business functions and compares the bottom-up results by KeyPro with the top-down perceptions of process managers. Finally, Sect. 6 concludes by outlining limitations and future research.

2 Conceptual Foundations on Process Importance

In agreement with the contributions by Davenport and Short [13] and Gibb et al. [5], we define a business process as an organized set of different activities which are performed in a logical sequence, and which consumes inputs to produce outputs in different forms. In addition, we define "process importance" as the degree to which a business process impacts the ability of the organization to create value, achieve organizational goals, and ultimately performance. Besides, we define a DSS as any system to address semi-structured or unstructured problems to support decision-making processes of users (e.g., [14, 15]).

As a conceptual foundation for KeyPro, this section introduces several dimensions of process importance which can be objectively quantified by existing data stored in information systems. In total, we identified four dimensions of process importance in current research. As process importance is a multidimensional construct, each dimension reflects a different perception of process importance. First, the *number of process executions* reflects the assumption that more important business processes are executed more often in an organization. Second, the *involved process stakeholders* dimension builds on the idea that important business processes are characterized by a higher number of people involved in process execution. Third, the *customer or supplier involvement* dimension is closely related to *value chain position* and assumes that business processes with a direct interface to the external environment such as customers or suppliers are more important for the organizational success. Fourth, *value*

chain position builds on the academic distinction into primary and secondary activities in the value chain as an objective classification into value and non-value generating activities. The following paragraph describes each of these dimensions in more detail and introduces quantifiable proxies for each dimension.

Number of Process Executions
Several academic contributions such as Tenhiälä [16] introduce volume as a measure of business processes. Thus, we introduce the number of executions of a business process as a first indication of potential process importance. This dimension accounts for the number of process executions as a volume construct and assumes a higher volume to be associated with higher importance (e.g., [16]). For example, products and services with higher demand and thus a higher performance impact require a higher number of process executions for production and demand satisfaction and different production systems (e.g., [17, 18]). We operationalize the number of process executions by the average number of process executions over a certain period of time to objectively measure the volume of a certain type of transaction or process execution.

Involved Process Stakeholders
As a second dimension of process importance, the number of people in organization involved in the execution of a business processes serves as an indication of how important a process might be for the organization [19]. People and people management are essential for process-oriented organizations, and thus serve as another indication of process importance. For example, the contribution by Yoon et al. [20] defines the degree of labor intensity as "the amount of people's time and effort necessary to solve the problem" [20]. We therefore operationalize *involved process stakeholders* by the average number of different process stakeholders involved in a business process for a certain period of time to proxy for process importance in this dimension.

Customer and Supplier Involvement
Third and in addition to *number of process executions* and *involved process stakeholders*, literature finds customer [21] and/or supplier influence [22] to be of high importance for business processes. For example, customer satisfaction potentially influences firm performance [23], and process failure becomes more damaging in presence of customers as customers might switch to another provider [24], which might negatively impact the lifetime value of customers [25]. In addition, the seminal contribution by Champy [26] highlights the requirement to consider business processes spanning across organizational boundaries to customers and suppliers up- and downstream in the value chain. Consequently, *customer and supplier involvement* therefore captures whether a business process has a direct interface to customers and/or suppliers in either a binary or value-weighted way.

Value Chain Position
Finally, we identified *value chain position* as a fourth dimension of process importance to account for the type of business process and the strategic position in the value chain of an organization. As indicated in exploratory workshops conducted in cooperation with our industry partner, decision-makers stated that managers tend to overestimate

the significance of their own process of responsibility for the organizational value creation and tend to classify own processes as primary activities. Therefore, we additionally classify business processes as primary or secondary activities along the widely accepted value chain by Porter [27] to gain an academic classification of whether a certain business process is directly or only indirectly related to the organizational value creation.

The frequently cited contribution by Ould [28] distinguishes processes into core processes such as the service of external customers, into support processes for the service of internal customers and the support of core processes, as well as management processes to administer the organization [5]. Likewise, the seminal contribution by Porter and Millar [29] distinguishes process into primary and secondary activities. Primary activities are comparable to core processes in Ould [28] to clearly distinguish the organization from competition. According to Duan et al. [30] primary activities are activities such as the physical creation, logistics, sales, and pre- and after-sales [30]. In contrast, secondary processes are similarly performed across organizations, thus adding little uniqueness to the particular organization [5, 27]. These supportive processes merely support primary activities by providing necessary inputs such as resources [30]. In sum, we introduce the distinction into "primary" processes directly related to the value creation and into "secondary" processes including management processes which are performed to support primary processes (Fig. 1).

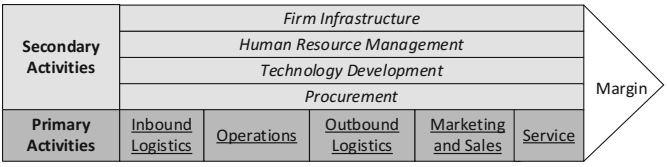


Fig. 1. Value chain by Porter [27] with primary and secondary processes

The dimension *value chain position* is therefore operationalized by assigning business processes into "primary" and "secondary" processes in Porter's concept of the organizational value chain [29].

3 Meta-Requirements and Design Principles

Based on the outcomes of the expert workshops in the problem awareness phase, we derived two meta-requirements (MRs) for KeyPro articulated by managers. As a first MR identified in the workshops, the solution should provide an objective measurement of process importance based on data stored in existing organizational information systems. In order to measure process importance, we consulted existing literature to identify dimensions of process importance and adjacent proxies which can be determined objectively from system data. Thus, we formulate MR1 as follows:

MR1: The artifact needs to extract source data from organizational information systems and compute process importance objectively.

Furthermore, the artifact is required to present findings in an interactive user interface which provides decision-makers the possibility to explore important business processes as an input for decision-making. Thus, we formulate MR2 as follows:

MR2: The artifact needs to provide decision-makers with the possibility to interactively explore important business processes.

Based on these two meta-requirements, we articulate design principles (DPs) for the artifact from a conceptual point of view. Thereby, we distinguish three layers: First, the source information systems layer comprises the different underlying information systems which provide the data necessary for bottom-up determination of important business processes (DP1). Second, the data management layer consolidates and transforms data retrieved from the organizational information systems layer and computes process importance (DP2). Third, the visualization layer presents the results to users and allows for interactive exploration (DP3). Figure 2 illustrates the conceptualization.

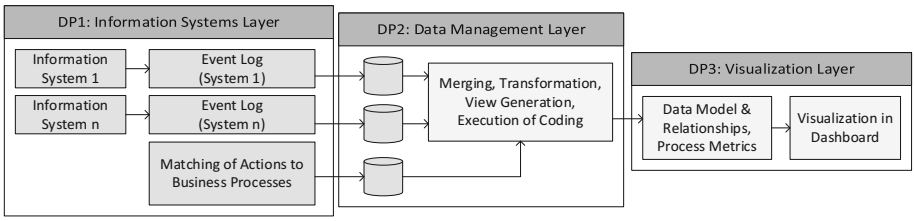


Fig. 2. Conceptualization of KeyPro

DP1: Provide Source Organizational Information Systems Layer

MR1 requires the ability to integrate data sources such as organizational information systems to measure proxies of process importance. The layer comprises log data stored by the IS and requires each entry to possess a timestamp for the exact determination of when the process was executed. Further, a unique identifier is required to assign an action in the system to a particular business process. In addition, the *involved process stakeholders* dimension requires information about the person or user who executed the action. Furthermore, to determine *customer and supplier involvement*, for each execution the information is necessary whether the action has a direct customer or supplier interface. Furthermore, each action performed in the IS needs to be assigned to either a primary or secondary business process for the *value chain position* dimension.

DP2: Provide Data Management and Process Importance Computation Layer

In addition, the data management layer is required to consolidate the data retrieved in the source organizational information systems layer and to merge data from different

sources of process information. For example, organizations frequently implement more than one information system to manage business processes. As process information is disseminated across various sources such as ERP, WfM, CRM, SCM or other systems, the layer needs to consolidate all data for a holistic overview over all systems. Additionally, the data management layer is required to transform the data for the calculation of the metrics of process importance identified in Sect. 2 and pre-compute process importance for all processes and sub-processes.

DP3: Provide Interactive Decision Support with Visualization Layer

Finally, the visualization layer needs to provide the possibility to interactively explore business processes along the dimensions of process importance as articulated by MR2.

4 Implementation

To build and evaluate KeyPro, we established an industry alliance with the IT service provider of a large German manufacturing corporation. The corporation consists of several sub companies with around 8.200 employees and 1.1bn Euro in turnover in 2017. The corporation is active in 23 countries with more than 50 locations and focuses on manufacturing products in B2B and B2C markets. Figure 3 illustrates the KeyPro prototype implementation from the IS layer to the final visualization layer based on the conceptualization in Sect. 3.

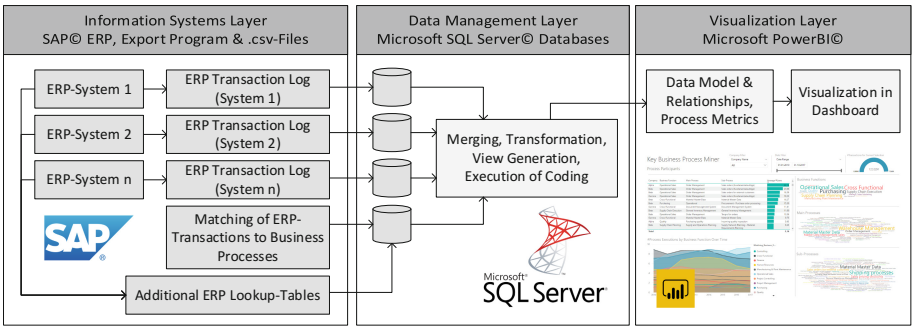


Fig. 3. Technical conceptualization

DD1: Source Organizational Information Systems Layer: SAP-ERP Systems

Our dataset comprises an event log from three real-world SAP ERP systems of the manufacturing corporation. The log file captures each transaction execution which results in a change to the underlying ERP database. An SAP transaction is a function or running program to perform certain actions in the ERP system [31]. Therefore, a transaction is comparable to a process step and can thus be assigned to one or more

Table 1. Overview over dataset from ERP systems (TA = Transaction)

ERP system	Total number of changes	Changes in database unrelated to transaction (TA) (removed)		Number of changes remaining	Unique TAs
Company A	42.666.436	10.206.791	23.92%	32.459.645	363
Company B	88.245.019	15.738.565	17.86%	72.506.454	582
Company C	22.035.778	1.497.374	6.80%	20.538.431	433
Total	**152.947.233**	**27.442.730**	**17.94%**	**125.504.530**	**773**

business processes. In addition, a change is defined as any change to a field in a data table due to a transaction execution.

In total, the event log file covers a period from 01/01/2010 to 31/10/2017 and includes 152.947.233 changes for all companies. We removed changes unrelated to transactions such as ERP-internal actions which are not due to the execution of business processes, leaving a final sum of 125.504.530 changes (executed process steps).

We implemented an ABAP table extractor application within the SAP ERP system to export relevant log data plus additional lookup-tables as .csv file close to real-time. The program can be implemented in any SAP ERP system. Therefore, KeyPro is able to handle and combine data from multiple SAP ERP systems, and to display results close to real-time. The following Table 1 gives an overview over the ERP log files.

Additionally, the information systems layer contains a lookup-table which contains the matching of transactions to the business processes. To determine the list of business processes for matching and the later evaluation, we consulted a pool of 52 process responsibles from four different companies at the industry partner to create a list of all main business processes and associated sub-processes in the corporation. Process owners are organized in a matrix, such that each process owner is responsible for one business function in one particular sub-company. Business functions include departments of the organizations such as “Finance”, “Operational Sales”, or “Manufacturing and Plant Maintenance”. Each business function further comprises several main processes such as “Order Management” in Sales. Each main process is further split into several sub-processes such as “Sales Orders for External Customers” in Order Management.

For each business function, process owners performed at least one workshop session of about 3 h to collect all main processes and adjacent sub-processes. Process owners created a total collection of 220 sub-processes distributed across 13 business functions and 49 main processes. Afterwards, 773 unique transactions of the SAP-ERP system were matched to this list of business processes retrieved from the process owners. Each SAP transaction was then mapped to a sub-process in the process list.

DD2: Data Management and Process Importance Calculation Layer: Microsoft SQL Server

All event log files are imported into a central database in the data management layer. Due to the wide dissemination in organizations and the free availability in the Express Edition, the data management layer is implemented using Microsoft SQL Server. However, KeyPro scripting can be run on any SQL-compliant relational database system. The data management layer further contains merging steps of the individual ERP files into one central database including data transformation and process importance computation steps.

DD3: Visualization Layer: Microsoft PowerBI

Finally, we implemented the visualization layer in Microsoft PowerBI© due to the capability to handle large amounts of data and the rich pageant of different visualizations, the free availability of the solution and its ability to connect to many different database formats. The visualization layer in Microsoft PowerBI contains one dashboard page for each of the dimensions as described earlier (Figs. 4 and 5).

Each dashboard provides the ability to filter by organizations to analyze and compare business processes across different companies. All dashboards such as the diagrams or word clouds in Microsoft PowerBI provide the ability to filter on the dashboard, page, and report level. Further, each dashboard page contains a time filter to analyze process importance and related metrics for a specific period of time and to analyze the evolution of process importance over time. For each level in the hierarchy of business functions, main- and sub-processes, a word cloud illustrates the most important processes (word size according to importance metrics). All fields contain the ability to select and filter for specific processes, and thus to drill-down into metrics for specific processes.

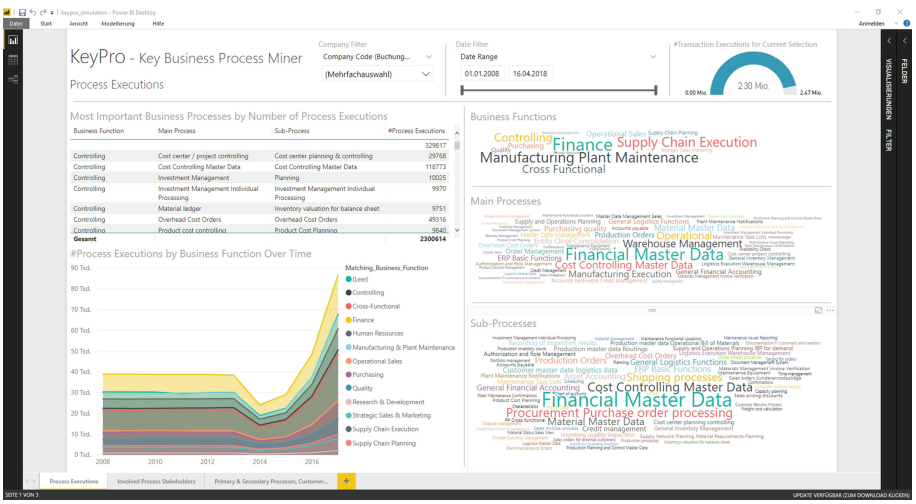


Fig. 4. Dashboard for *number of process executions (2010–2017)*

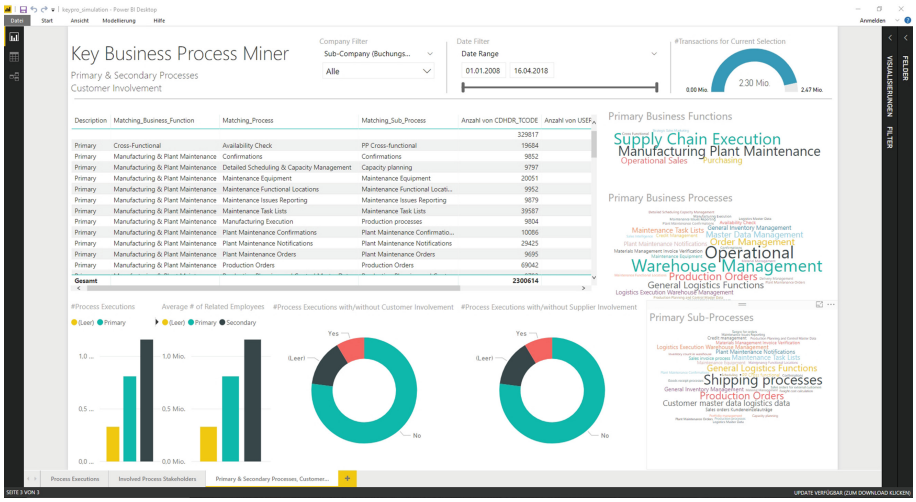


Fig. 5. Dashboard for *involved process stakeholders (2010–2017)*

5 Evaluation

We performed a field evaluation with two case studies to validate and scrutinize the potential of KeyPro to enrich process understanding of organizational decision-makers. The evaluation of KeyPro was performed in two case studies for two business functions in company "C", namely "Controlling" (including 6 main processes with 14 different sub-processes), and "Finance" (including 10 main processes with 18 different sub-processes). Decision-makers were asked to determine the following metrics in Table 2 for the dimensions of process importance derived in Sect. 2 for each sub-process in their area of responsibility. As several metrics such as the number of process executions, or the number of stakeholders related to a business process are time-dependent and potentially changing over time, process owners were asked to provide an average over the last 12 months. Process owners were further asked to also include system-performed actions in their indication to account for processes not triggered explicitly by stakeholders themselves.

5.1 Case Study 1: Evaluation for Business Function "Controlling"

For the business function "Controlling", process owners indicated a mean number of 80.03 sub-process executions per month (min = 0, max = 400, SD = 134.61).

According to the top-down evaluation by process owners, the two most important main processes for Controlling in the *number of process executions* dimension are "Product Cost Controlling" (n = 460 mean monthly executions) and "Profitability Analysis" (n = 326 mean monthly executions), while "Material Ledger" and "Financial Details" constitute the least important main processes (with n = 0.17 and n = 0.25 mean monthly executions, respectively). Within the most important main process

Table 2. Top-down evaluation by process owners (exemplary excerpt)

Business function	Main process	Sub-process	Avg. executions	Avg. distinct users	Customer	Supplier	Value chain position
Controlling	Investment management	Planning & administration	33	1	No	No	Secondary
Finance	Accounts payable	Accounts payable management	2.000	2	No	Yes	Secondary

"Product Cost Controlling", the sub-processes "Product Cost Planning" was indicated at n = 400 executions, "Calculation of Intercompany Prices" at n = 30 executions, and "Update Transfer Price Data" at n = 30 executions. In the *involved process stakeholders* dimensions, process owners indicated a mean of 1.82 distinct persons involved in the execution of a sub-process (min = 0, max = 5, SD = 1.64). Regarding stakeholders, "Profitability Analysis" constitutes the most important main process with a mean number of 10 different people being involved per month, while "Product Cost Controlling" constitutes the second most important main process with n = 7 stakeholders. Further, as expected for the internal Controlling business function, none of the sub-processes was indicated to have any direct interface to customers or suppliers in the s*upplier or customer involvement* dimensions. Finally, in accordance with the scientific allocation of "Controlling" to the secondary activities in the value chain by Porter [27], process owners perceived all main and sub-processes as secondary to the value creation.

Compared to results delivered bottom-up by KeyPro, significant differences are revealed for the preceding 12 months prior to the evaluation. In *number of process executions*, the most important main process is "Material Ledger" with n = 32.150.25 monthly executions (due to the automatic inventory valuation for balance sheets performed by the ERP). The second-most important main process is "Cost Center & Project Controlling" with a mean of 505.92 monthly executions. This strongly contrasts top-down perceptions. For example, the "system reality" for "Product Cost Controlling" reveals only n = 36.5 monthly executions of the associated ERP transaction, such that process owners significantly overestimate the importance of this process. In terms of *involved process stakeholders*, KeyPro revealed the main processes "Product Cost Planning" (n = 1.20 stakeholders) and "Cost Center & Project Controlling" (n = 1.11 stakeholders) to be most important.

5.2 Case Study 2: Evaluation for Business Function "Finance"

In addition to "Controlling", we performed a second evaluation for the business function "Finance". Process owners reported a mean of 1019.06 sub-process executions per month (min = 0, max = 5500, SD = 1716.61). In the perception of process owners, the two most important main processes in *number of process executions* are "Accounts Payable" (n = 9.500 mean monthly executions) and "Accounts Receivable and Credit Management" (n = 5.750 mean monthly executions). This perception strongly

contrasts bottom-up findings by KeyPro, with the main process "Financial Master Data" and "Documentation FI Customers" being the most executed process (n = 2.221.92 and n = 426 mean monthly executions, respectively). The main processes "Accounts Payable" and "Accounts Receivable and Credit Management" were executed much more infrequently than stated by process owners (n = 344.42 and n = 426 mean monthly executions, respectively). Furthermore, process owners stated that the main processes "Incoming Payments" and "Foreign Currency Management" were not executed by the department at all. However, still the associated ERP transactions were executed several times during the year preceding the evaluation at n = 0.08 and n = 1.33 times a month on average, which indicates the processes were "forgotten" by process owners in workshops due to their infrequency.

In terms of the *involved process stakeholders* dimension, process owners stated the main processes with the highest number of distinct stakeholders likewise being "Accounts Payable" and "Entity Close and Consolidation" with n = 9 and n = 8.17 different stakeholders, respectively. This perception is partly revoked by findings from ERP data in KeyPro. Although "Accounts Payable" is executed by the highest number of distinct users and thus can be termed the most important main process in accordance with top-down perceptions, the absolute number of different stakeholders executing the process strongly contrasts managerial perceptions. While managers believe the number of different process executors related to "Accounts Payable" is at an average of n = 9, the corresponding ERP transactions are executed by only 1.89 stakeholders per month. The same finding holds true for "Entity Close and Consolidation", with only 1.07 different stakeholders being involved in transaction executions. Furthermore, KeyPro ranks the main process "Accounts Receivable and Credit Management" as the most important process in the department in terms of different stakeholders with a monthly average of 1.92 different users. Regarding *customer or supplier involvement*, managers stated the sub-processes "Credit Management", "Dispute Management", and "Accounts Payable Invoice Management" to have a direct interface to customers. These perceptions are supported by KeyPro, however, KeyPro in addition highlights the sub-process "Dunning Run" to have a customer interface, which was not considered by process owners. Regarding supplier involvement, both human managers and KeyPro in accordance find "Accounts Payable" to be the only process having a supplier interface. However, for the sub-process of incoming paper-based invoices, KeyPro was unable to detect the process in the log data, and thus was outperformed by human managers. Finally, in terms of *value chain position*, all process owners perceived their processes as secondary to the organizational value creation in accordance with the value chain by Porter [27].

6 Conclusion

Organizational decision-makers in practice often do not have a precise understanding of which of the organizational business processes are of prime importance for the organization. At the same time, prioritization of process decision-making is a critical element in business process management activities such as transformation projects, resource allocation, or process monitoring. At the same time, existing literature in

academics does not provide an exhaustive answer to a definition of process importance and to the question of how the importance of processes can be measured objectively from different, partially contradictory viewpoints.

Addressing these research gaps, this paper proposes a design science approach to develop the decision support system "KeyPro" to automatically discover and explore important business processes from organizational information systems. In addition to the prototype implementation, this paper further suggests a conceptualization of process importance and measurable proxies motivated from existing literature. Dimensions of process importance include the number of process executions, involved process stakeholders, customer and/or supplier involvement, and value chain position. For each dimension, we suggest an operationalization of process importance based on existing log data to measure process importance in information systems.

Furthermore, we retrieved a collection of 220 business processes from a pool of 52 process experts across four companies at an industry partner. Overall, 773 ERP transaction programs were matched to the business processes to objectively determine which business processes constitute important processes along the dimensions identified in literature. KeyPro was then applied in a real-world case involving three independent SAP ERP systems of a manufacturing corporation. In the evaluation phase, we compared KeyPro-based important process suggestions with importance perceptions by process owners. Our evaluation demonstrates the potential of the data-driven approach as it reveals significant deviations between top-down importance perceptions by process owners and bottom-up results suggested by KeyPro.

6.1 Limitations and Future Research

This work encounters several limitations and directions for future research. First, we had to retrieve a specific list of business processes from the different sub-companies at the industry partner to be able to conduct the later evaluation. The processes represent a rather specific view of the companies. To increase generalizability, KeyPro should be able to match transactions against reference models such as the APQC Process Framework [32] in future. Second, another limitation refers to the inability of KeyPro to integrate off-system processes into solution finding. Although one manager highlighted the strong reliance of the company on the ERP system with "the overwhelming share of processes being implemented into the SAP system", KeyPro is thus far only able to determine business processes from information systems. However, the data-driven approach does not take into account paper-based or off-system processes such as the development of vision and strategy from the APQC framework, or managerial processes (e.g., [32]). However, in principle KeyPro is able to handle any log file which contains a timestamp, a transaction or process identity, as well as information about the dimensions of process importance such as the user performing the process. Thus, the implementation should also include other information systems. Third, we developed the artifact based on data from three manufacturing companies. As opposed to service organizations with a higher share of intangible and non-repetitive business processes, the reliance on manufacturing with a large share processes being executed and recorded in information systems makes KeyPro more targeted towards industrial organizations, while service organizations might experience a higher

implementation effort. Fourth, an additional limitation refers to the mapping of ERP transactions to the process list, which was performed by the authors of this paper. With regard to further research, we plan to match the transactions with another focus group of process responsibles in a 1:n manner to account for transactions which occur in multiple business processes. Sixth, the proxies used for determining process importance may be enhanced. Thus, we plan to test and evaluate more elaborated metrics in the future. For example, *customer and supplier involvement* can be determined by whether the ERP transaction accesses data related to a customer or supplier in the ERP. We further plan to determine automatically whether a transaction is related to a customer or supplier.

References

1. Teece, D.J.: Business models, business strategy and innovation. Long Range Plan. **43**(2–3), 172–194 (2010)
2. Afflerbach, P., Leonhard, F.: Customer experience versus process efficiency: towards an analytical framework about ambidextrous BPM: completed research paper. In: Thirty Seventh International Conference on Information Systems (2016)
3. Piccinini, E., Gregory, R.W., Kolbe, L.M.: Changes in the producer-consumer relationship - towards digital transformation. In: Wirtschaftsinformatik Proceedings, pp. 1634–1648 (2015)
4. Lucas Jr., H.C., Agarwal, R., Clemons, E.K., El Sawy, O.A., Weber, B.: Impactful research on transformational information technology: an opportunity to inform new audiences. Manag. Inf. Syst. Q. **37**(2), 371–382 (2013)
5. Gibb, F., Buchanan, S., Shah, S.: An integrated approach to process and service management. Int. J. Inf. Manag. **26**(1), 44–58 (2006)
6. van der Aalst, W., et al.: Business process mining: an industrial application. Inf. Syst. **32**(5), 713–732 (2007)
7. Gopal, R., Marsden, J.R., Vanthienen, J.: Information mining—reflections on recent advancements and the road ahead in data, text, and media mining. Decis. Support Syst. **51**(4), 727–731 (2011). Recent Advances in Data, Text, and Media Mining & Information Issues in Supply Chain and in Service System Design
8. Caron, F., Vanthienen, J., Baesens, B.: Comprehensive rule-based compliance checking and risk management with process mining. Decis. Support Syst. **54**(3), 1357–1369 (2013)
9. Fischer, M., Heim, D., Janiesch, C., Winkelmann, A.: Assessing process fit in ERP implementation projects: a methodological approach. In: Maedche, A., vom Brocke, J., Hevner, A. (eds.) DESRIST 2017. LNCS, vol. 10243, pp. 3–20. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59144-5_1
10. Hayashi, A.M.: Thriving in a big data world. MIT Sloan Manag. Rev. **55**(2), 35 (2014)
11. Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: A framework for efficiently mining the organisational perspective of business processes. Decis. Support Syst. **89**, 87–97 (2016)
12. Kuechler, B., Vaishnavi, V.: On theory development in design science research: anatomy of a research project. Eur. J. Inf. Syst. **17**(5), 489–504 (2008)
13. Davenport, T.H., Short, J.E.: The new industrial engineering: information technology and business process redesign. Sloan Manag. Rev. **31**(4), 11–27 (1990)
14. Shim, J.P., et al.: Past, present, and future of decision support technology. Decis. Support Syst. **33**(2), 111–126 (2002)

15. Sprague, R.H.: A framework for the development of decision support systems. Manag. Inf. Syst. Q. **4**(4), 1 (1980)
16. Tenhiälä, A.: Contingency theory of capacity planning: the link between process types and planning methods. J. Oper. Manag. **29**(1–2), 65–77 (2011)
17. Kim, Y., Lee, J.: Manufacturing strategy and production systems: an integrated framework. J. Oper. Manag. **11**(1), 3–15 (1993)
18. Schroeder, D.M., Congden, S.W., Gopinath, C.: Linking competitive strategy and manufacturing process technology. J. Manag. Stud. **32**(2), 163–189 (1995)
19. Willaert, P., Van den Bergh, J., Willems, J., Deschoolmeester, D.: The process-oriented organisation: a holistic view developing a framework for business process orientation maturity. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 1–15. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_1
20. Yoon, Y., Guimaraes, T., Clevenson, A.: Exploring expert system success factors for business process reengineering. J. Eng. Technol. Manag. **15**(2–3), 179–199 (1998)
21. Chase, R.B.: The customer contact approach to services: theoretical bases and practical extensions. Oper. Res. **29**(4), 698–706 (1981)
22. Yoo, S.H., Shin, H., Park, M.-S.: New product development and the effect of supplier involvement. Omega **51**, 107–120 (2015)
23. Anning-Dorson, T.: Customer involvement capability and service firm performance: the mediating role of innovation. J. Bus. Res. **86**, 269–280 (2018)
24. Hess Jr., R.L., Ganesan, S., Klein, N.M.: Service failure and recovery: the impact of relationship factors on customer satisfaction. J. Acad. Mark. Sci. **31**(2), 127–145 (2003)
25. Kumar, V., Petersen, J.A.: Using a customer-level marketing strategy to enhance firm performance: a review of theoretical and empirical evidence. J. Acad. Mark. Sci. **33**(4), 504–519 (2005)
26. Champy, J.: X-Engineering the Corporation: Reinventing Your Business in the Digital Age. New Warner, San Antonio (2002)
27. Porter, M.E.: The Competitive Advantage: Creating and Sustaining Superior Performance. Free Press, New York (1985)
28. Ould, M.A.: Business Processes: Modelling and Analysis for Re-Engineering and Improvement. Wiley, Chichester (1995)
29. Porter, M.E., Millar, V.E.: How information gives you competitive advantage (1985)
30. Duan, C., Grover, V., Balakrishnan, N.R.: Business process outsourcing: an event study on the nature of processes and firm valuation. Eur. J. Inf. Syst. **18**(5), 442–457 (2009)
31. Orosz, T. (ed.): Analysis of SAP development tools and methods. IEEE (2011)
32. American Productivity & Quality Center (APQC): Process Classification Framework (PCF). https://www.apqc.org/pcf. Accessed 18 Nov 2017

Tell Me What's My Business - Development of a Business Model Mining Software

Visionary Paper

Christian Fleig(✉), Dominik Augenstein, and Alexander Maedche

Information Systems and Service Design (ISSD),
Karlsruhe Institute of Technology, Institute of Information Systems
and Marketing (IISM), Karlsruhe, Germany
christian.fleig@kit.edu

Abstract. Decision-making, innovation and transformation of business models require organizations to precisely understand their current status-quo business model. However, traditional top-down approaches to business model visualization are human-centric, "de jure", subjective, error-prone, and time-consuming. In contrast, bottom-up Business Model Mining collects and consolidates different sources of business model-related data and presents business models in predefined templates such as the Business Model Canvas. This paper employs a design science approach to develop a Business Model Mining Software to automatically mine business models bottom-up from enterprise resource planning systems. First, this paper discusses weaknesses of traditional top-down approaches to business model visualization and the potential by data-driven approaches. Second, this paper derives meta-requirements and design principles as theoretical foundations to conceptualize the business model mining software. Third, this paper instantiates the "Business Model Miner" in Microsoft PowerBI as a working software based on data from three real-life SAP R/3 ERP-systems of a manufacturing corporation.

Keywords: Business Model Mining · Business Model Canvas
Enterprise Resource Planning (ERP) Systems · Design science

1 Introduction

Organizations navigate in environments characterized by a multitude of changes in business, technologies, and individuals (e.g., [1]). Exemplary "megatrends" such as the "Big Data Era", Globalization, the "Internet of Things", Big Data, Social Media, Connectivity, the Sharing Economy, Individualization, or Mobility trends require organizations to continuously reinvent themselves and their business models through the provision of new products, services, or a combination of both. Besides competitive necessities to redefine the organizational value creation and to satisfy customer demands to remain competitive (e.g., [2]), organization science with the seminal contingency theory by Donaldson [3] highlights the need to adapt the organizational design including business models to match both internal and external variables in response to environmental change.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 105–113, 2018.
https://doi.org/10.1007/978-3-319-92901-9_10

However, decision-making in business model innovation and transformation requires a precise understanding of the current status-quo business model. This paper follows the definition proposed by Osterwalder and Pigneur [4] and defines a business model as a description of "the rationale of how an organization creates, delivers, and captures value". To visualize business models, academia and practice developed a rich pageant of different modelling methods, techniques, and tools to visualize business models (e.g., [5]). As a core obstacle in business model transformation, these traditional top-down approaches are subjective, human-centered, and "de jure" by relying on manual inputs and tacit knowledge of individual decision-makers. Thus, traditional top-down approaches to business model creation are expensive and time-consuming in their creation, error-prone, superficial, and potentially suffer from biases to the "system reality". In addition, the prevailing de jure approaches such as the widely accepted "Business Model Canvas" (BMC) by Osterwalder [6] provide rather high-level and strategic inputs, which are less focused on the actual operationalization of the business model [7] and thus only of limited use in business model innovation and transformation. Further, to increase the value of business modelling, strategic inputs need to be connected to the operational layer such as business processes (e.g., [7, 8]).

To overcome these outlined weaknesses, the "Big Data Era" provides significant potential for bottom-up and data-driven approaches to improve business modelling and decision-making. In his contribution, van der Aalst et al. in [9] coined the term "Mine Your Own Business" and proposed to use process mining and big data for decision making. As an extension, we aim to contribute to this proposal by using big data to mine business models of organizations. For example, new technologies such as data- or process mining allow to complement traditional top-down approaches to business modelling (e.g., [10]). Besides, enterprise resource planning (ERP) systems play a major role in organizations [11]. For example, ERP-systems such as the SAP Business Suite or Oracle are information systems are implemented by organizations to increase automation and to integrate business processes and information within one common infrastructure [12]. Organizational information systems including ERP-, Supply Chain Management (SCM), Customer Relation Management (CRM), or Workflow Management (WfM) systems generate large amounts of business-related data during operations (e.g., [13]), and cover the entire range of business processes in the organization [14]. Thus, such data can be used in business model mining [15] comparably to other data mining approaches such as process mining [16]. With the term "Business Model Mining" [15], we refer to approaches to mine business models from information systems and to visualize them in templates such as the BMC. Thereby, Business Model Mining has the potential to enrich and to supplement top-down, "de jure" business modelling approaches with bottom-up, data-driven and "de facto" knowledge on business models. In sum, the research question of this paper becomes:

RQ: "How to design a business model mining software to retrieve business models bottom-up from data stored in information systems?"

To answer the research question, this paper employs a design science approach to design the software artifact of the "Business Model Miner" to mine a BMC automatically from enterprise resource planning (ERP) systems such as the SAP R/3

Business Suite. To the best of our knowledge, the artifact in this paper is the first software being able to retrieve a BMC from ERP-systems.

The remainder of this paper is organized as follows. Section 2 briefly introduces the design science research methodology. Section 3 presents meta-requirements for business model mining and derives design principles to develop the conceptualization for business model mining. Section 4 presents the instantiation of the "Business Model Miner" in Microsoft SQL and Microsoft PowerBI for SAP R/3 ERP-systems. Section 5 concludes by presenting limitations and avenues for future research.

2 Research Methodology

Design science research aims to systematically conceptualize and develop artifacts in order to solve real-world problems [17]. To develop the "Business Model Miner", we employ a design science research (DSR) approach based on the methodology proposed by Kuechler and Vaishnavi [18] and consists of a problem awareness, suggestion, development, and an evaluation phase. In the initial problem awareness phase of design cycle one, we conduct a series of discovery workshops at the industry partner of this research to discover the weaknesses of de jure approaches to business modelling and the arising need for bottom-up business model mining. In the suggestion phase, we derive a set of preliminary meta-requirements for the business model mining concept in preceding research [15], which we refine in this paper with adjacent design principles and design decisions. In the development phase, we instantiate the conceptualization in a working prototype of the "Business Model Miner". Finally, as each of the approaches is associated with particular strengths and weaknesses, we hypothesize that the combination of both human, tacit knowledge on business models together with data-driven inputs from information systems significantly improves business modelling. However, design science requires at the forefront to solidly evaluate the Business Model Miner artifact in terms of whether the software actually provides superior or complementary inputs. Thus, we evaluate the Business Model Miner at three companies with a pool of thirty department managers by comparison of top-down, tool-based, and mixed workshops and individual sessions. Thus, in the evaluation phase, the prototype will be evaluated in terms of the ability to complement or outperform traditional approaches by comparison against "golden standard" business models derived in expert workshops with department responsibles at the industry partner.

3 Meta-Requirements and Design Principles

We derive three meta-requirements (MRs) and associated design principles (DPs) for business model mining in research preceding the development of the final "Business Model Miner" in this paper. This section refines these preliminary MRs and DPs identified in the previous contribution by Augenstein and Fleig [15]. We enhance the meta-requirements based on a series of workshops with IT and business model experts from the IT service provider of the industry partnership underlying this research.

First, business model mining requires data from information systems and the software artifact needs knowledge on which data provides the relevant inputs for which of the building blocks of the business model. Thus, MR1 demands:

MR1: "To mine business models bottom-up from information systems, business model-related data needs to be identified and retrieved".

Second, business model mining extracts large amounts of data from various sources such as multiple ERP-systems, which store business model-related data across numerous data tables. In addition, organizations frequently have more than one ERP-system in place. Thus, these different data sources need to be consolidated and prepared for later visualization of the business model in the Canvas and for analysis. Thus, MR2 imposes the following requirement on Business Model Mining:

MR2: "To mine business models from information systems, different sources of business model-related data need to be consolidated."

Third, the business model needs to be represented in a uniform template such as the BMC for the algorithms to be able to retrieve and visualize business models. Therefore, MR3 requires:

MR3: "To mine and analyze different business models from information systems, business models need to be visualized in a predefined template."

Based on these three meta-requirements, we introduce three associated design principles. First, to be able to retrieve the building blocks of the BMC, the relevant data tables in the ERP-system need to be identified, extracted, and connected via primary keys. Thus, business model-related data in one or more ERP systems is identified and extracted in individual files to account for MR1 in a "ERP-Systems Layer". We formulate the first design principle accordingly:

DP1: "Business Model Mining requires an ERP-systems layer to extract and identify relevant data"

Second, business model data needs to be merged in one central database and preparatory steps and scripting needs to be performed to account for MR2. Thus, the "Data Consolidation, Scripting, and Preparation Layer" merges for later visualization of the business models in the canvas. We formulate the second design principle as follows:

DP2: "Business Model Mining requires a database layer to consolidate and prepare business model-related data"

Third, MR3 requires the visualization in a predefined template. In particular, the BMC by Osterwalder [4] has gained significant popularity in both academia and practice. In general, the goal of business model tools is to provide a complete, transparent, and easy-to-understand representation of the business model [15, 19]. We formulate the final design principle as:

DP3: "Business Model Mining requires a presentation layer to present business models in a predefined business model template and provide additional analysis functionality"

Thus, the "Business Model Presentation Layer" visualizes the business model in the canvas template and provides additional functionality such as the calculation of indicators, time and company filters, or word clouds. The Canvas contains 9 building blocks which try to capture and structure the business model into a predefined template. Therefore, the Canvas serves as the template for the later surface of the Business Model Miner. For each of the building blocks, we introduce one or several proxies which reflects the definition of the building block and which can be computed from data stored in information systems in accordance with the contribution by Osterwalder and Pigneur [4]. "*Customers and Suppliers*" captures people and organizations targeted by the business model. Further, "*Value Propositions*" comprises the products and services through which the business model creates value for the customer segments. "*Channels*" collects the different ways of how organization communicates and how the value propositions are delivered to customers. "*Customer Relationships*" defines the type of customer relationships such as customer retention and acquisition. "*Revenue Streams*" gathers the different types of revenue generated by the business model. "*Key Resources*" represents the assets which are vital for the business model. "*Key Activities*" is the building block of the canvas which captures the "most important actions" which need to be done. "*Key Partnerships*" comprises the pool of suppliers and partners in the value chain of the organization to enable the business model through resource acquisition (e.g., purchasing materials). "*Cost Structure*" captures the most important expenditures incurred for carrying out the business model. Table 1 contains an overview over the proxies chosen for each building block.

Table 1. Proxies from information systems for building blocks of the BMC

BMC building block	Proxies ("design decisions")
Key activities	Matching of executed transactions in the ERP-system ("event log") to the APQC process framework [20] and counting number of executions, the number of human users related to process execution, the involvement of a customer or supplier in the transaction, or the value of sales or purchases linked to the transaction
Value proposition	Amount and value of products and services sold (product groups and hierarchies)
Customer relationships	Value generated with customers; repeat buying/single transactions; duration of customer relationships
Channels	Amount and value of products and services sold over distribution channels
Customer segments	Customer industries; customer classifications; geographic customer information
Revenue structure	Main revenues from balance sheets
Key resources	Most highly valued tangible and intangible assets
Cost structure	Main expenditures from balance sheets
Key partners	Supplier industries; geographic supplier information; duration of supplier relationships

In sum, these meta-requirements and design principles serve as the theoretical guidelines during the actual development of the Business Model Miner. The following paragraph presents results from the implementation of the Business Model Miner, which retrieves data from an SAP-ERP system (DP1), consolidates and prepares data in a Microsoft SQL Server database (DP2), and visualizes the BMC in Microsoft PowerBI (DP3). Figure 1 illustrates the blueprint of the technical implementation of the Business Model Miner based on the meta-requirements and design principles.

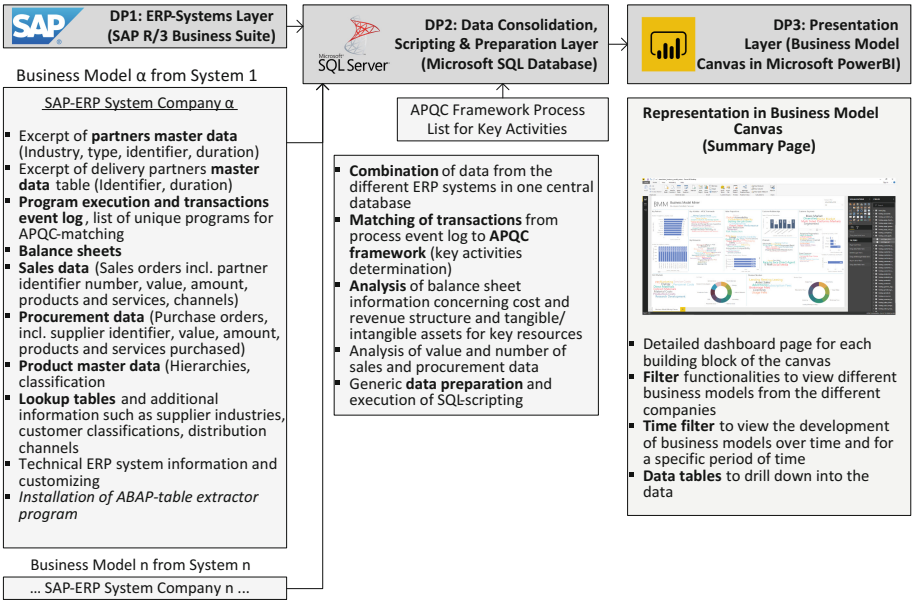


Fig. 1. Blueprint conceptualization and implementation of the business model miner for one or multiple SAP R/3 ERP systems and visualization of the BMC in Microsoft PowerBI

4 The Business Model Miner

We implement the instantiation of the Business Model Miner based on data from three real-world SAP R/3 ERP-systems from a German manufacturing corporation. The corporation consists of five individual companies with about 8,200 employees in 22 countries and 45 locations. The corporation is active in business-to-business and business-to-customer markets of various industrial areas and achieved a turnover of about 1.2 billion Euro in 2016. We retrieved business model mining data for three sub-companies for a period between 2010 and 2017. Each company is implemented on one SAP system, such that the business model of the particular company can be distinguished along the boundaries of the respective SAP systems.

For each building block of the Canvas, the tool presents word clouds and diagrams. The size of the tags in the word clouds is scaled according to values such as sales or

purchase values or numbers such as the volume of products sold or purchased. Users can adjust the level of details and specify the number of elements to be displayed in the word clouds and dashboards (e.g., the top N for each of the proxies). Besides, the screen contains a company to filter to select the business model of one or more individual companies. Further, the date filter allows to select business models over a specific period of time. Each of the visualization dashboards provides the ability of Microsoft PowerBI to filter distinct elements and associated data. For each of the building blocks, an additional detailed analysis dashboard page with further visualizations and drill-down possibilities is provided. Figure 2 contains the instantiation in Microsoft PowerBI based on a randomization of values for reasons of company compliance.

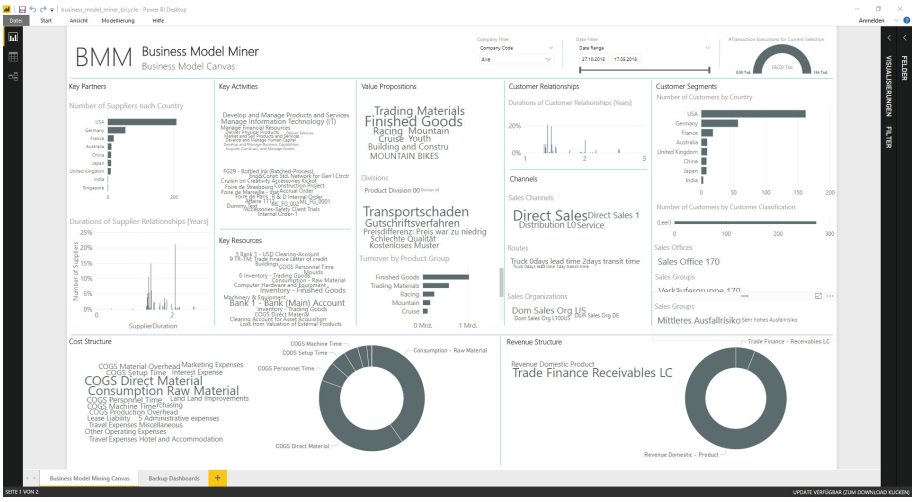


Fig. 2. Business model canvas dashboard of the business model miner *(randomized values)*

5 Conclusion and Outlook

This paper proposes the "Business Model Miner" to mine a BMC from SAP R/3 ERP-systems and to improve transformation decision-making in the "Big Data Era". However, the approach to derive business models bottom-up from data in ERP systems also encounters several limitations. First and comparable to "shadow processes" in process mining, business model mining captures only the elements of the business model which are stored or executed in information systems. Therefore, business model mining fails to include business model-related elements outside of systems such as paper-based processes, or intangible parts of the value proposition which are not documented in systems. Second, organizations might have more than one business model. In future research, we aim to explore how to distinguish among different business models in one ERP system. As a take-away from these limitations, we

position business model mining as a bottom-up "stimulus" to enrich and to complement the traditional top-down, human-centered approaches. Business Model Mining complements rather than replaces traditional "de jure" business modelling techniques with a "de facto" and data-driven approach to retrieve the business model automatically from information systems. Based on this Forum paper, we aim to provide the community with both an innovative Business Model Mining concept and the actual implementation in a working piece of software. The state of research of this paper also paves the way for future research avenues. The "mining" capabilities of the tool can be improved by means of artificial intelligence to replace some of "reporting" functionality with more elaborate business model discovery techniques. Finally, our SAP table extraction program allows to export data close to real-time. Thus, we aim to provide another version of the Business Model Miner to support "Real-Time Business Model Mining".

References

1. Loebbecke, C., Picot, A.: Reflections on societal and business model transformation arising from digitization and big data analytics: A research agenda. J. Strateg. Inf. Syst. **24**, 149–157 (2015)
2. Piccinini, E., Gregory, R.W., Kolbe, L.M.: Changes in the producer-consumer relationship-towards digital transformation. In: Wirtschaftsinformatik (2015)
3. Donaldson, L.: The Contingency Theory of Organizations. Sage, Thousand Oaks (2001)
4. Osterwalder, A., Pigneur, Y.: Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. Wiley, Hoboken (2010)
5. Ebel, P., Bretschneider, U., Leimeister, J.M.: Leveraging virtual business model innovation: A framework for designing business model development tools. Inf. Syst. J. **26**, 519–550 (2016)
6. Osterwalder, A.: The Business Model Ontology: A Proposition in a Design Science Approach (2004)
7. Di Valentin, C., Burkhart, T., Vanderhaeghen, D., Werth, D., Loos, P.: Towards a Framework for Transforming Business Models into Business Processes (2012)
8. Bonakdar, A., Weiblen, T., Di Valentin, C., Zeißner, T., Pussep, A., Schief, M. (eds.): Transformative influence of business processes on the business model: classifying the state of the practice in the software industry. IEEE (2013)
9. van der Aalst, W.: Mine your own business: using process mining to turn big data into real value. In: ECIS 2013 Completed Research (2013)
10. Veit, D., Clemons, E., Benlian, A., Buxmann, P., Hess, T., Kundisch, D., Leimeister, J.M., Loos, P., Spann, M.: Business models. Bus. Inf. Syst. Eng. **6**, 45–53 (2014)
11. Fischer, M., Heim, D., Janiesch, C., Winkelmann, A.: Assessing process fit in ERP implementation projects: a methodological approach. In: Maedche, A., vom Brocke, J., Hevner, A. (eds.) DESRIST 2017. LNCS, vol. 10243, pp. 3–20. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59144-5_1
12. Gattiker, T.F., Goodhue, D.: What happens after ERP implementation: understanding the impact of interdependence and differentiation on plant-level outcomes. MIS Q. **29**, 559–585 (2005)
13. Hayashi, A.M.: Thriving in a big data world. MIT Sloan Manag. Rev. **55**, 35 (2014)
14. Ehie, I., Madsen, M.: Identifying critical issues in enterprise resource planning (ERP) implementation. Comput. Ind. **56**(6), 545–557 (2005)

15. Augenstein, D., Fleig, C.: Exploring design principles for a business model mining tool. In: International Conference on Information Systems (2017)
16. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**, 1–37 (2013)
17. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. Manag. Inf. Syst. Q. **28**, 75–105 (2004)
18. Kuechler, B., Vaishnavi, V.: On theory development in design science research: anatomy of a research project. Eur. J. Inf. Syst. **17**, 489–504 (2008)
19. Kley, F., Lerch, C., Dallinger, D.: New Business Models for Electric Cars—A Holistic Approach. Energy Policy **39**, 3392–3403 (2011)
20. APQC: Process Classification Framework (PCF) (2017). https://www.apqc.org/pcf

Checking Business Process Correctness in Apromore

Fabrizio Fornari[1(✉)], Marcello La Rosa[2], Andrea Polini[1], Barbara Re[1], and Francesco Tiezzi[1]

[1] Univeristy of Camerino, Camerino, Italy
{fabrizio.fornari,andrea.polini,barbara.re,francesco.tiezzi}@unicam.it
[2] University of Melbourne, Melbourne, Australia
marcello.larosa@unimelb.edu.au

Abstract. In this paper we present the integration of BProVe - Business Process Verifier - into the Apromore open-source process analytics platform. Given a BPMN model BProVe enables the verification of properties such as soundness and safeness. Differently from established techniques for BPMN verification, that rely on the availability of a mapping into a transition based formalism (e.g. Petri Nets), BProVe takes advantage of a direct formalisation of the BPMN semantics in terms of Structural Operational Semantics rules. On the one side, this still permits to give precise meaning to BPMN models, otherwise impossible due to the usage of natural language in the BPMN standard specification. On the other side, it permits to overcome some issues related to the mapping of BPMN to other formal languages equipped with their own semantics (e.g. non local effects of BPMN elements such as termination). The defined BPMN semantics has been implemented in MAUDE. Through the MAUDE Linear Temporal Logic (LTL) model checker, correctness properties encoded in LTL formulae are evaluated and the result is then presented to the user. The integration into Apromore allows model designers to use the Apromore BPMN editor to design models and to interact with BProVe to verify properties of the designed model. The results are shown graphically on top of the process model, so as to highlight behavioural paths that violate the correctness properties. Designers can then easily identify the violation and repair the model accordingly.

1 Introduction

The Business Process Model and Notation (BPMN 2.0) [15] is a widely accepted language for modelling business processes, which has been standardised by the Object Management Group in 2011. A BPMN model, like any other process model, should be easy to understand [4] and correct [5] to maximise its usefulness and uptake. Specifically, a process model is correct if it does not exhibit structural issues, such as disconnected nodes, or behavioural ones, such as soundness, safeness and other, possibly domain dependent, behavioural anomalies. In particular, recognising correctness problems as soon as possible helps to avoid

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 114–123, 2018.
https://doi.org/10.1007/978-3-319-92901-9_11

negatively impacting effects before the process model is deployed. This is particularly important if the model is intended to be executed on top of a Business Process Management system, where undetected problems may lead to expensive manual interventions to rectify business operations (e.g. aborting an inconsistent order, recovering a double payment, or restoring a loan application that got stuck). Several properties have been defined to guarantee process correctness such as the ones reported in [9,18,19].

A prominent correctness property is that of *soundness*. Informally, a model is sound if and only if it satisfies the following three rules: (i) *Option to complete*: requiring that a process instance should always complete, once started; (ii) *Proper completion*: requiring that when a process ends there should not be any other activity still running; (iii) *No dead activities*: requiring that a process does not contain activities that will never be executed.

Safeness is another relevant property that can be verified over business process models. If safeness is verified, this guarantees that no activity in a process model will ever be executed in more than one instance concurrently. Indeed a multiple concurrent execution of the same activity is perceived as a bad behaviour that could lead to unwanted effects such as e.g. multiple payments or multiple deliveries of the same product.

Additionally, more specific properties can be defined in order to ensure the conformance of the model to specific behavioural patterns. For example, this may be useful to check the correct exchange of messages as well a proper evolution of process activities (we will refer to them in this paper as *domain dependent* properties).

Armed with the objective of building high-quality BPMN models through an easy to use modelling and analysis environment, we have developed a novel tool for Business Process Verification (BProVe) [5,6] packaged as a RESTful web service. In this paper, we present the integration of BProVe into Apromore[1] [11], an open-source process analytics platform developed by the BPM community. Apromore was originally conceived as a process model repository, but over time it evolved into a BPM platform that offers various types of analytics, along the various phases of the BPM lifecycle (discovery, analysis, redesign, implementation and monitoring). The platform is based on a service-oriented architecture, and deployed as a Software as a Service. Apromore has been conceived has an easy extensible framework, where new plugins can be added to an ecosystem of advanced business process analytics capabilities. BProVe's ability to perform a comprehensive correctness check of BPMN models, together with the possibility of highlighting the elements that led to the violation of a given correctness property, make BProVe a valuable addition to the Apromore platform. In particular, this integration enables Apromore users to increase their trust on the quality of the models stored in the platform's repository.

Existing tools for process model verification mostly focus on Petri nets/Workflow nets (e.g. WoPeD) or offer support for the core set of BPMN elements only (i.e. excluding elements with non-local semantics such as OR-joins).

[1] http://apromore.org.

The latter is achieved via an internal transformation into Petri nets (e.g. ProM 5.2) or other formalisms, though these transformations are only partial in nature, often limited by the target language such as Petri net. In addition, these tools are typically available offline. An exception is Signavio Academic Initiative, which provides basic correctness checking for single-pool BPMN models. In contrast, our tool potentially supports the full BPMN language as it relies on a direct semantics rather than on an internal transformation, and is available online, as a Software as a Service.

The rest of this paper is organised as follows. Section 2 provides an overview of BProVe with details on the BProVe architecture, its features, and discusses the results of a large experiment against several thousand models. Section 3 describes the integration of BProVe into Apromore, with a focus on the user interface. Finally, Sect. 4 provides links to a screencast and to the tools' Web pages.

2 BProVe - Business Process Verifier

BProVe - Overview. BProVe allows to automatically verify relevant properties for business process models. Differently from similar tools, BProVe is based on a *native BPMN semantics* [8] defined according to a Structural Operational Semantics style [16], which makes it suitable for a MAUDE implementation [2]. MAUDE is a programming language that models systems, and the actions within those systems. It can be used to define executable formal models of distributed systems, and it provides analysis tool to formally analyse the models. In MAUDE, a distributed system is formalised as a rewriting logic theory [3,12]. Implementing the BPMN Operational Semantics in MAUDE has the great benefit of ensuring the faithfulness of the implemented semantics with respect to its theoretical definition, as operational rules are directly implemented as MAUDE rewriting rules. In addition, due to its direct BPMN Operational Semantics, BProVe *re-conducts verification results* to the original model, so that diagnostic information can be easily reported on the model in a way that is understandable by process stakeholders. This is especially useful when many parties with different background need to quickly interact based on models.

BProVe - Architecture. The architecture of the BProVe toolchain is illustrated in the component diagram of Fig. 1 which consists in a quite standard organisation of components for providing a verification as a service. From the left side of the figure to the right, we can see three main components: Modelling Environment, BProVe WebService, and BProVe Framework. The Modelling Environment component represents the tool used to design BPMN models and to specify which properties of the model the user wants to verify. It is here that the integration with Apromore takes place. Apromore's BPMN editor is used for the design of BPMN models. By means of a BProVe plug-in for Apromore, a user can pass from the design phase to the verification phase still remaining inside the modelling environment. The BProVe plug-in constitutes what in

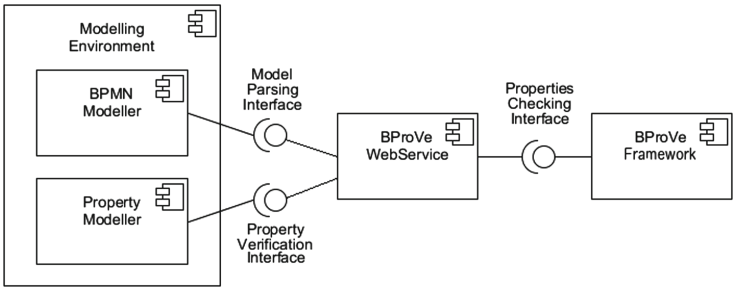


Fig. 1. Component diagram of the BProVe toolchain.

the diagram is referred with the terms Model Parsing Interface and Property Verification Interface. This is used to reach the BProVe WebService, and to interact with it via HTTP requests. A BPMN model and a property to verify are sent to the WebService, which handles the requests by parsing the BPMN model and the property into the syntax accepted by the BProVe Framework. The parsing of a BPMN model results in a model represented by means of the BPMNOS Syntax while the result of parsing the property is an LTL formula. The BProVe Framework component is based on a running instance of MAUDE loaded with the MAUDE modules implementing the BPMN Operational Semantics and the LTL MAUDE model checker [10].

BProVe - Features. BPMN 2.0 is a quite complex modelling notation including around one hundred graphical elements, even though some of them are not really used in practice [14]. BProVe permits to easily handle a wide set of BPMN elements (see [5]). In particular, it is possible to consider BPMN elements that can have non-local effects that are difficult to handle with tools transforming BPMN models to Petri Nets. For instance, the handling of *termination end events*, that permit to quickly abort a running process, is usually not supported due to the inherent complexity of managing non-local propagation of tokens in Petri Nets; this feature is instead natively supported by the semantics at the base of BProVe. However, extending the framework to include further elements is not particularly challenging (even a problematic element as the OR-join can be conveniently added to our formalisation, see [7]).

Moreover, the main motivation to use Petri Nets encodings is the availability of already developed tools supporting verification [13]. However, such tools generally work well for standard Petri Nets, but they do not support extensions necessary to encode BPMN-specific features such as the management of task state evolution (e.g., enabling, running and complete). BProVe instead is based on an extensible framework that is able to potentially support all the features of BPMN, as the approach permits to apply language extensions to cover new features without affecting the verification phase.

A further advantage of BProVe is its support for BPMN *Collaborations*. This enables the analysis of inter-organisational correctness, which is still not

supported by most of the available tools [1]. Results of checking safeness and soundness over BPMN collaborations differ from results obtained through encodings, which usually introduce a mapping at process level, and then compose the processes in a collaboration by means of an inner transition. This often results in imposing an a priori upper bound on the number of pending messages [17].

BProVe - Maturity. BProVe was already tested, as reported in [5], over 1026 models coming from the BPMN Academic Initiative (http://bpmai.org/). More recently, after some updates aiming at fixing parsing functionalities, we run a new massive verification session over the same dataset with the aim of verifying soundness and safeness properties. We are now able to handle 2569 models (see Table 1).

Table 1. Fraction of models satisfying soundness.

Property	Models satisfying it
Soundness	1279 (49%)
Option to Complete	1745 (67%)
Proper Completion	2380 (92%)
No Dead Activities	1937 (75%)
Safeness	2428 (94.5%)

The results of the session tell us that out of the 2569: 1745 models respect the *Option to Complete* property (67% of the total models); 2380 models respect the *Proper Completion* property (92% of the total models); 1937 models respect the *No Dead Activities* property (75% of the total models). In conclusion, 1279 models were detected as sound (49% of the total) which means those are the models that respect all the three properties mentioned above. In relation to safeness 94.5% models respect the property.

For a deeper insight of the observed models we refer to data reported in Table 2. The table reports a classification of models based on the number of BPMN elements they present. For each class we reported the total number of models belonging to that class and the percentage of models that satisfy the analysed properties. It is true that the majority of the analysed models are composed of less than 20 elements, and that those models are possibly designed by "junior" practitioners, and they are not coming from industrial organisations. However, one can reasonably argue that if soundness problems emerge with models that have a relatively low and manageable number of elements, they will most certainly emerge with more complex models, which have a higher number of elements.

Then, we collected in Table 3 data on the average time required to verify properties. Overall, we can observe that, as it can be expected, times increase with the dimension of the models even if it is still manageable.

Table 2. Fraction of models satisfying the considered properties (in class).

Class	Number of models	Option to complete %(models)	Proper completion %(models)	No dead activities %(models)	Safeness %(models)
0 − 10	1530	79% (1216)	96% (1475)	81% (1241)	97% (1489)
11 − 20	808	57% (461)	89% (724)	69% (558)	93% (751)
21 − 30	183	31% (57)	85% (155)	57% (105)	87% (159)
31 − 40	33	18% (6)	57% (19)	70% (23)	64% (21)
41 − 100	13	38% (5)	54% (7)	77% (10)	61% (8)

Table 3. Average time (in ms), clustered in classes according to number of elements in the model.

Class	Option to complete avg. time (ms)	Proper completion avg. time (ms)	No dead activities avg. time (ms)	Safeness avg. time (ms)
0−10	252	220	59	375
11−20	2737	1937	1283	3687
21−30	670	4559	4480	2422
31−40	7061	7219	15137	6493
41−100	5660	1412	59738	7455

Summing up, the extended validation confirms the results we reported in [5], such as it is not seldom to find models that violate relevant behavioural properties, also after their release. In addition the experiments confirm that our approach seems to be applicable in practice to realistic BPMN models.

3 BProVe Plugin for Apromore

The BProVe tool has been integrated into Apromore in the form of a plugin for the "*Editor*" environment. The Apromore Editor allows users to create and edit BPMN models. Our plugin can verify correctness properties over these models. The result is an integrated editor that hides the complexity of verification to the user, favouring then the habit of verifying models for correctness before their release. In fact, as discussed in the previous section, it is not seldom the case that models are approved and released without much attention in relation to quite basic correctness properties.

When accessing BProVe from the editor, the BProVe Web service is invoked with the BPMN model open in the editor. The editor then waits for the model parsed into the BPMN Operational Semantics Syntax to be returned. At the same time, the "Check model correctness with BProVe" menu pops up. This will enable the selection of the properties to verify, and then to display the

results. The properties mainly correspond to a predefined set of LTL templates. This choice was made to ease the usage of verification techniques by non-expert users. That said, one is assumed to be familiar with basic behavioral anomalies of process models, such as deadlocks, livelocks and dead activities, to understand the output provided by our tool. As future work, we envision a couple of improvements: (i) a visual animation of the behavioral anomalies identified by the tool (e.g. a deadlock) to lower the entry bar for non-expert users, and (ii) a second interface to allow expert users to enter arbitrary LTL expressions corresponding to properties of their interest.

A screenshot of the Apromore Editor with the BProVe menu is shown in Fig. 2. The checking menu is mainly divided into three parts described in the following.

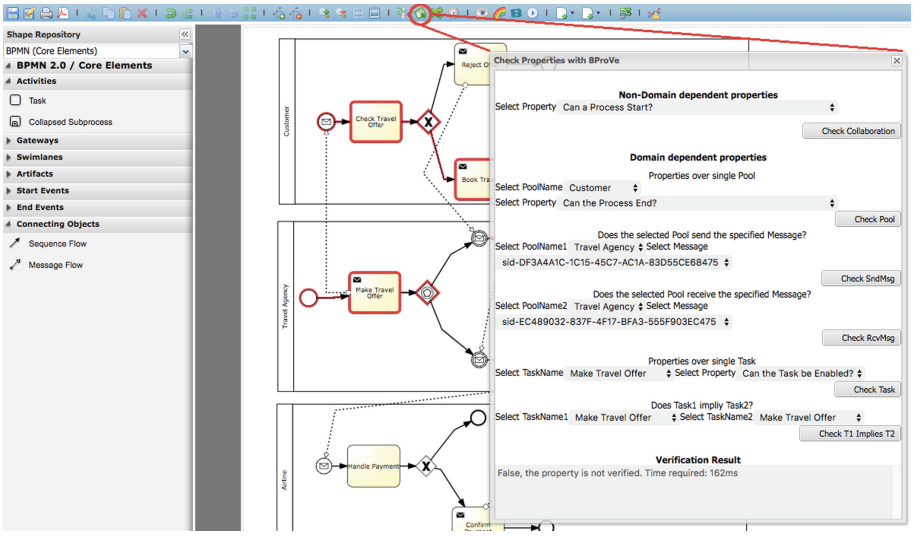


Fig. 2. BPMN model correctness checking with BProVe in Apromore.

Non-domain Dependent Properties. It reports a drop-down menu which permitting to select properties that will be checked over the process models within the collaboration, as well as on the whole collaboration. This property checking functionality does not require any other specification from the user. The properties are here shortly detailed.

- "*Can a Process Start?*" It verifies whether at least a process, in the collaboration, is able to start.
- "*Can a Process End?*" It verifies whether at least a process, in the collaboration, is able to reach the end state.
- "*Can All the Processes End?*" It verifies whether all the involved processes will eventually reach the end state.

- "*No Dead Activities*" It verifies that no Task in any process of the collaboration is dead. A dead task is a task that will never be executed.
- "*Option to Complete*" It verifies that all the processes involved in the collaboration will reach the end state, once started.
- "*Proper Completion*" It verifies for all the processes in the collaboration, that once a process reaches the end state, it does not have any token still circulating. This allows verifying that no other Task will be executed once the process has reached the end state.

Domain Dependent Properties. It reports several drop-down menus which permit to the user to select specific components of the collaboration (Pools, Tasks, and Messages) and to run verification over those components. This part of the menu can be seen as divided into three sub-parts here shortly detailed.

- The "*Verification of properties over a single Pool*" part allows one to select a Pool from a drop-down menu, reporting the list of pools present in the collaboration, and to select one of the, previously described, "Non-Domain dependent properties" that makes sense verifying over a specific Pool. All the properties that can be verified over the entire collaboration can be verified for a specific Pool except for the one that checks whether all the Processes of a collaboration can end.
- The "*Verification of Message exchange*" part allows to select Pools and Messages and to check whether a specific Pool will Send a specified Message, or to check whether a specific Pool will Receive a specified Message.
- The "*Verification of Task execution*" part allows to select a Task and to check whether it will reach the status "Enabled", "Running" or "Completed". It also allows to verify whether the execution of a Task implies the execution of another one, by selecting that task and requesting a check.

Verification Result. It reports a text area used first to display the parsed model from the BPMN notation to the BPMN Operational Semantics Syntax, returned from a first call of the BProVe Web service, then to display the results of each property verification (returned from any other call of the BProVe Web service). The result of a property verification can be "True" or "False" plus the addition of the time, in millisecond, required to process that verification. In addition, a counterexample is available which allows highlighting also in the modelling environment the elements that lead to the property violation (see elements with the red border in Fig. 2).

4 Screencast and Links

A screencast is available at https://youtu.be/lFiS_Y-gygQ. This video illustrates a typical scenario where the user requires to check correctness properties of a BPMN model. BProVe is an open source software; it can be redistributed and/or modified under the terms of the GPL2 License. BProVe source code, as well as instructions for using it, can be found at http://pros.unicam.it/tools/bprove.

The BProVe Apromore integration is embedded as a set of OSGi plugins into the online process analytics platform Apromore, which has been used for the screencast (http://apromore.org).

References

1. Breu, R., Dustdar, S., Eder, J., Huemer, C., Kappel, G., Köpke, J., Langer, P., Mangler, J., Mendling, J., Neumann, G., Rinderle-Ma, S., Schulte, S., Sobernig, S., Weber, B., Towards living inter-organizational processes. In: 15th IEEE Conference on Business Informatics, pp. 363–366 (2013)
2. Clavel, M., Durn, F., Eker, S., Lincoln, P., Mart-Oliet, N., Meseguer, J., Talcott, C.: All About Maude-A High-Performance Logical Framework: How to Specify, Program and Verify Systems in Rewriting Logic. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
3. Clavel, M., Eker, S., Lincoln, P., Meseguer, J.: Principles of maude. Electron. Notes Theor. Comput. Sci. **4**, 65–89 (1996)
4. Corradini, F., Ferrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., Spagnolo, G.O.: A guidelines framework for understandable BPMN models. Data Knowl. Eng. **113**, 129–154 (2018)
5. Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F., Vandin, A.: BProVe: a formal verification framework for business process models. In: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, pp. 217–228. IEEE Press (2017)
6. Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F., Vandin, A.: BProVe: tool support for business process verification. In: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, pp. 937–942. IEEE Press (2017)
7. Corradini, F., Muzi, C., Re, B., Rossi, L., Tiezzi, F.: Global vs. local semantics of BPMN 2.0 OR-join. In: Tjoa, A.M., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 321–336. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_23
8. Corradini, F., Polini, A., Re, B., Tiezzi, F.: An operational semantics of BPMN collaboration. In: Braga, C., Ölveczky, P.C. (eds.) FACS 2015. LNCS, vol. 9539, pp. 161–180. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28934-2_9
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, 2nd edn. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-642-33143-5
10. Eker, S., Meseguer, J., Sridharanarayanan, A.: The maude LTL model checker. Electron. Notes Theor. Comput. Sci. **71**, 162–187 (2004)
11. La Rosa, M., Reijers, H.A., Van Der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., GarcA-Bauelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. **38**(6), 7029–7040 (2011)
12. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theor. Comput. Sci. **96**(1), 73–155 (1992)
13. Morimoto, S.: A survey of formal verification for business process modeling. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008. LNCS, vol. 5102, pp. 514–522. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69387-1_58

14. zur Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69534-9_35
15. OMG: Business Process Model and Notation (BPMN V 2.0) (2011)
16. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebr. Program. **60**(61), 17–139 (2004)
17. van der Aalst, W.M.P.: Structural characterizations of sound workflow nets. Comput. Sci. Rep. **96**(23), 18–22 (1996)
18. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28616-2
19. Wynn, M.T., Verbeek, H.M.W., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Business process verification-finally a reality!. Bus. Process Manag. J. **15**(1), 74–92 (2009)

Aligning Goal and Decision Modeling

Renata Guizzardi[1], Anna Perini[2], and Angelo Susi[2(✉)]

[1] Ontology and Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo, Vitória/ES, Brazil
rguizzardi@inf.ufes.br
[2] Fondazione Bruno Kessler, Trento, Italy
{perini,susi}@fbk.eu

Abstract. Making decisions is part of the everyday life of any organization. Documenting such decisions is important to allow organizational members to learn from past mistakes, and to support newcomers in adjusting into the organization. However, nowadays, decisions are usually documented in unstructured ways, which makes it hard to find and interpret them. In this work, we propose the alignment of goals and decision modeling, by using two existing modeling notations, namely *i** and DMN. This alignment is based on a semantic approach, in which the elements of the notations are mapped into a decision-making ontology, also developed in the context of this work.

Keywords: Decision-making · Goal · Alignment of modeling notations Ontology

1 Introduction

Decision-making is an essential part of the everyday life of any organization. Individuals working in several points of action of the organization need to make decisions in a daily basis in order to accomplish their goals. In this context, documenting decisions is useful for: (a) capturing expert knowledge on decision-making to help understand how decisions have been made, possibly avoiding pitfalls i.e. decisions that led to unwanted outcomes; (b) providing newcomers with the means to understand how the decisions have been made, so that they can learn from them; and (c) documenting the evolution of organizational services and products in terms of the decisions that have been made in their regard.

Organizations usually document decisions by recording them as lessons learned [1]. However, these records are often made in natural language, thus being generally unstructured. Recently, OMG has created the Decision Modeling Notation (DMN) [2], aiming at providing a standard language to model human decision-making, and to model and implement automated decision-making. Using a modeling language to document decisions may prove useful to allow a more consistent, structured and uniform view about decisions taken within an organization.

Most of the works applying DMN use this language in connection to the Business Process Modeling Notation (BPMN), as a means to make explicit the specific tasks where decisions are taken [2, 3]. However, in these cases, much of the rationale is

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 124–132, 2018.
https://doi.org/10.1007/978-3-319-92901-9_12

hidden w.r.t. 'why' a decision must be taken (i.e. which goals it achieves) and 'why' using a specific set of criteria to take a decision. Providing an in depth analysis of this rationale is exactly the focus of goal-oriented modeling. Hence, we claim that aligning DMN with a goal-oriented modeling language will allow one to make explicit the decision rationale. Specifically, we consider the *i** framework [4], since it is among the most used ones in goal-oriented modeling research, and it supports several language extensions, but our approach is general and can be adapted to other languages.

In this short paper, we present the key idea in our approach, that is the alignment of *i** and DMN, using an ontological approach to map the elements of each notation. By doing so, we aim at making sure that the alignment is based on the real semantics of the notation's elements, not only on their label. This approach has been followed with success in previous works [5, 6]. To support the alignment, we propose a decision-making ontology, whose development is based on the Unified Foundational Ontology (UFO) [7, 8]. We choose UFO because it already covers some basic concepts that are needed to define the decision-making domain.

Throughout the years, there have been some related works supporting decisions' representation and reasoning. For example, the works about decisions on goals in KAOS [9] and in the Business Intelligence framework [10]. Lamsweerde [9] proposes the use of decision tables to represent the options related to goal achievements in an OR goal decomposition. The tables allow to perform both qualitative and quantitative reasoning about the best choice of goals achievement, given some quality criteria to be ensured. The Business Intelligence (BI) framework [10] proposes the use of GRL modelling language (based on *i**) to perform a quantitative reasoning on goal achievements, thus supporting the decisions related to the best strategies to pursue a given business objective. Both these proposals add elements (such as tables, or business parameters) either as tools not integrated or as extension of the modelling language (in the case of the BI proposal). On the contrary, our proposal intends to bridge and align two complementary languages using ontologies.

The remaining of this paper is organized as follows: Sect. 2 presents the proposed alignment between goals and decision modeling; Sect. 3 illustrates the approach with a working example; and Sect. 4 concludes this paper.

2 Towards the Alignment of Goal and Decision Modeling

2.1 Decision-Making Ontology

Figure 1 shows the proposed ontology. As aforementioned, the ontology has been created based on a foundational ontology named UFO. Thus, the UFO concepts are labeled with *"UFOx:"* where "*x*" is substituted with the part of the foundational ontology which defines such concept. From now on, for clarity purposes, the words corresponding to the ontology concepts have a different font.

Before going into the concepts of the decision-making ontology, it is important to understand some UFO distinctions. The most fundamental distinction is the one between `Substantial` and `Event`. `Substantial` is a kind of Endurant, i.e. an entity that does not have temporal parts, and persists in time while keeping its identity

(e.g. a person and the color of an apple). On the other hand, Event is composed of temporal parts (e.g. heart attack, trip). A Substantial is also different from a Moment. While the former is an independent entity (e.g. a person or a car), the latter (usually referred to as property) is dependent on a Substantial (e.g. someone's headache, the brand of a car). We usually say that a Substantial bears a Moment, or that a Moment inheres in a Substantial [7]. An Action is a kind of Event performed by a particular type of Substantial, named Agent (examples of agents are person, student and software developer). Besides performing actions, agents can perceive events and bear special kinds of moments, named Mental Moments. Mental Moments are properties existing in the Agent's mind [8]. The remaining concepts of UFO are explained in the context of the decision-making ontology.

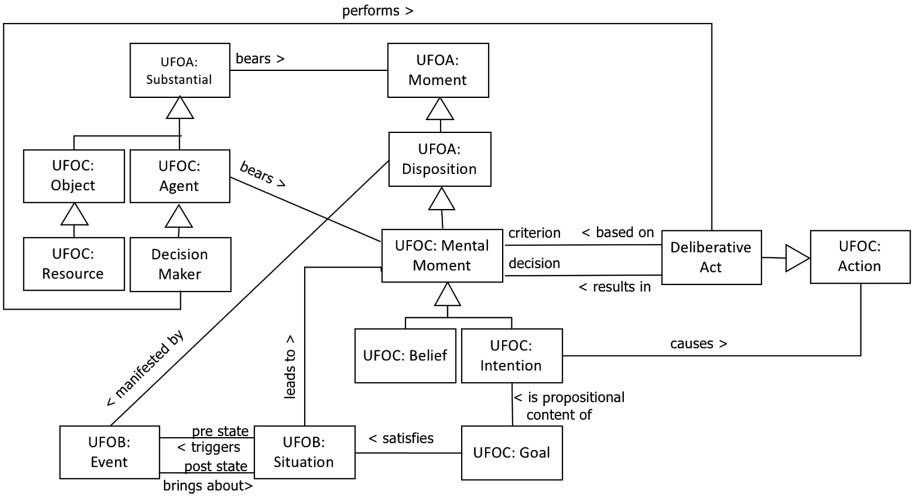


Fig. 1. Decision making core ontology conceptual model

A Decision Maker is an Agent who performs Deliberative Acts. A Deliberative Act is, on its turn, a kind of Action that results in the creation of a Mental Moment termed Decision. The Decision is thus beared by the Decision Maker. In other words, a Decision is a Mental Moment inhering in a Decision Maker, as a result of this Decision Maker's Deliberative Act.

As a Mental Moment, a Decision may be both a Belief or an Intention. For example, if by looking at a restaurant's menu, someone decides to order fish, then this Decision is an Intention. UFO defines Intention as a kind of Mental Moment associated to a particular Goal (i.e. the person's goal of ordering fish) and causing a particular Action (i.e. the person's action of ordering fish). On the other hand, if just by thinking or talking to others, one decides that one likes fish better than red meat, without actually acting on it, this Decision is a Belief, even if this may have an impact on the person's future intentions.

When performing a Deliberative Act, a Decision Maker may use some criteria to take a Decision. Thus, we say Deliberative Act is based on Criterion. A Criterion is also a kind of Mental Moment, i.e. a Belief or an Intention of the Decision Maker. For instance, taking the previous example, the Decision of ordering fish might have been taken based on one's Belief that fish is healthier than meat or in one's Intention to pay a more economic meal. Note that as both Criterion and Decision are Mental Moments, a Criterion may also be a previous Decision, thus allowing the creation of a chain of dependence between different decisions.

Understanding the consequences of a particular Decision is also an important issue in decision making. This may be achieved by analyzing the UFO concepts of Disposition, Event and Situation. A Disposition is a kind of property (i.e. Moment) of a Substance that may be manifested through the occurrence of Events. Take for example the Disposition of a magnet to attract metallic material. This magnet has this Disposition even if it is never manifested, for example, because the magnet is never close to any magnetic material [11]. A Mental Moment is a kind of Disposition and as such, it may be manifested by the occurrence of an Event. UFO defines Situation as a state of the world which may either trigger an Event (pre state) or be brought about by one (post state). The Decision's Consequence is the Situation brought about by the Event manifested by the Decision.

An interesting aspect of this ontology is to allow reasoning about a typical cycle in decision making processes. Based on specific criteria, a Decision Maker takes a Decision through a Deliberative Act. Such Decision is manifested by an Event which brings about a particular Consequence. Such Consequence leads to new Mental Moments (i.e. beliefs and intentions) in the mind of that Decision Maker, who is liable to take a different Decision next time, based on these revised Mental Moments (i.e. criteria).

2.2 Mapping Rules from *i** to DMN

This work takes a model-driven approach, in which *i** modeling elements are mapped into DMN modeling elements. By doing that, we aim at facilitating automatic mapping from one language to the other, whenever possible. Besides, we take into account the semantics of the modeling elements of each notation, mapping them to the decision-making ontology described in Sect. 2.1. An *i** modeling element corresponding to the same ontological concept of a DMN element will thus be mapped to each other. By taking this semantic strategy, we aim at avoiding the pitfall of mapping modeling elements only based on their label, which may be misleading.

A goal in *i** may be directly mapped to UFO Goal, which is basically defined as a proposition that satisfies a specific Situation (or as previously explained, a state of the world). As seen in Sect. 2.1, by possibly being an intention, a Decision has a goal as propositional content. This justifies the first mapping we make between ***i* goal and DMN decision***. It is important to emphasize that we are particularly interested in goals that are parents in OR-refinements, as the analysis of alternatives in *i** clearly indicates a point in which a decision needs to be made, i.e. the decision to pursue one

of the subgoals of the OR-refinement relation. The relation between ***goals that pertain to the same OR-refinement tree indicate a DMN decision dependency***. As seen in Sect. 2.1, this dependency happens because the `Consequence` of one `Decision` leads to the definition of criteria for another.

A resource in *i** may be either mapped to a `Resource` or to an information believed by an `Agent` (and thus a propositional content of a `Belief`). In case of the former, the *i** resource is mapped into a DMN knowledge source. In DMN, a knowledge source may be either a person or a document associated to a knowledge model containing the decision's criteria; or a person who participates in a decision [2]. Thus, we highlight that ***an i* resource is a DMN knowledge source document*** (not a person).

An actor in *i** is associated to `Agent`. ***The actor whose goals are being analyzed as well as the ones on whom this actor depend are mapped to DMN knowledge sources*** (however, human knowledge sources this time, not documents).

A quality goal in *i** is mapped to `Softgoal`, which is a particular kind of `Goal` in UFO, whose satisfaction is a relation connecting `Softgoal`, `Agent`, `Belief` and `Situation`. To be more precise, a `Softgoal` is said to satisfy a `Situation` according to the `Belief` of the `Agent` bearing an `Intention` whose propositional content is that `Softgoal`. A `Criterion` may be such `Intention`. In practice, this justifies the mapping between ***i* quality goal and DMN criterion***. The DMN elements explained in the previous paragraphs belong to a specific model named Decision Requirement Diagram (DRD). Nevertheless, a criterion belongs to a different model, known as Decision Table, which is associated to the DRD by being linked to a knowledge model. ***A knowledge model is thus also added to a DRD, inferred from the presence of quality goals in the i* model***.

Table 1 summarizes the mapping rules between *i** and DMN, also showing their associated UFO concepts.

Table 1. *i** to DMN mapping rules

i **Element**	**DMN Element**	**Ontological Concepts**
OR-refinement parent goal	decision	`Goal`
goals pertaining to the same OR-refinement tree	decision dependency	`Goal, Deliberative Act, Decision, Criterion`
resource connected to goal pertaining to an OR-refinement tree	knowledge source (document)	`Resource`
	input data	`Belief propositional content`
actor A whose rationale is being analyzed or other actors on whom A depends.	knowledge source (human)	`Agent`
quality goal (sometimes hardgoal) related to a goal pertaining to an OR-refinement tree	criterion to be refined in a decision table	`Goal (Softgoal), Intention, Criterion`
	knowledge model	`Criterion`

3 Working Example

Figure 2 depicts an *i** model of our working example. This model is adapted from a well-known case and it was chosen because it depicts all elements we need to illustrate the proposed alignment approach.

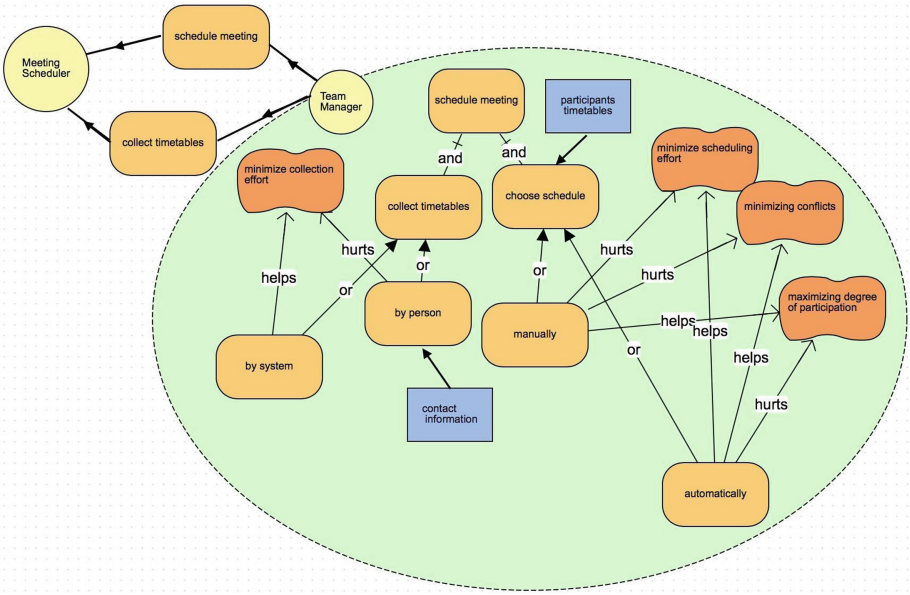


Fig. 2. *i** model of the Meeting Scheduling case

The model in Fig. 2 analyses the goals of a Team Manager actor, who wants to schedule a meeting. To accomplish such goal, s/he needs to collect the timetables from the participants and to choose a schedule. Then, the model shows some alternatives s/he may follow to accomplish her goals, which gives us the opportunity to explain how the *i** elements map to DMN elements. The resulting DMN DRD is depicted in Fig. 3.

The Team Manager needs to decide whether to collect timetables by herself or by the Meeting Scheduling system (see the goals *by person* and *by system* in Fig. 2). Thus, *collect timetables* (as a parent goal in an OR-refinement, as stated on Table 1) becomes a decision in the DMN DRD; for similar reasons, a decision has been added in the DRD for the *choose schedule* goal (refer to *Collect Timetable Mode* and *Choose Schedule Mode* decisions in Fig. 3).

Besides the decisions automatically added by following the mapping rules, we also noted that the goals *(choose schedule) manually* and *automatically* require decision-making. We thus added them to the DRD of Fig. 3, and we use colors to differentiate between modeling elements automatically inferred from the mapping rules of Table 1 (in white) from the ones we added later in the diagram (in grey).

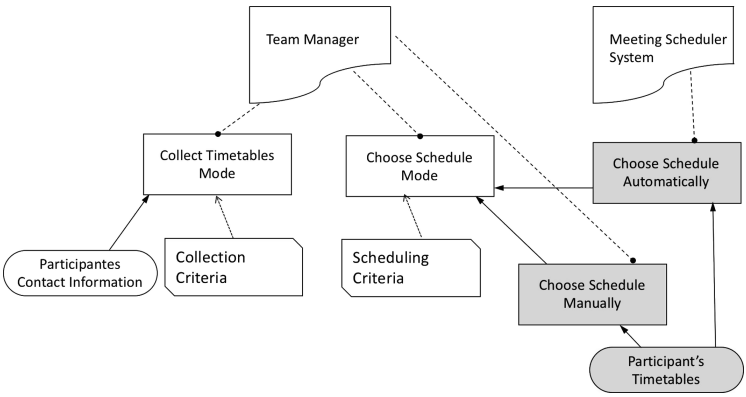


Fig. 3. DMN decision requirement diagram

The *Participant's Contact* input info of Fig. 3 comes from the *contact information* resource associated to the *(collect timetables) by person* goal of Fig. 2. Analogously, we added an input info entitled *Participant's Timetables*, which is related both to the *Choose Schedule Manually* and to the *Choose Schedule Automatically* decisions.

The actors in Fig. 2 became knowledge sources in the DRD of Fig. 3. The relation between the Team Manager and the decisions for which she is responsible may also be automatically inferred from Fig. 2, since the goals that originated such decisions belong to the perspective of the Team Manager. This is not true for the link between Meeting Scheduler and the Choose Schedule Automatic decision, manually added to the DRD.

Finally, the two DRD knowledge models may be inferred from the fact that there are a few quality goals related to the subgoals of the OR-refinement depicted in the goal model of Fig. 2. According to DMN, each knowledge model from a DRD is associated to a Decision Table, detailing the criteria used to take the decision. For example, Table 2 shows the Decision Table associated to the Collection Criteria knowledge model. The Decision Table's last column present the decision (e.g., manual and automatic) obtained by the analysis of the criteria's values; the first column enumerates the values to be analyzed; and the remaining columns present the criteria used to obtain this decision (in Table 2, only one criterion, i.e. the number of participants in a meeting).

Table 2. Example of DMN decision table

Collection criteria		
U	Number of participants	Collection mode
1	<=2	Manual
2	>2	Automatic

The criteria presented in the Decision Table are based on the quality goals of the *i** model. For instance, on Table 2, we interpret that the collection effort (based on the *minimize collection effort* quality goal) may be given by the number of participants in the meeting being schedule.

4 Conclusion

In this paper, we proposed the alignment between goal and decision modeling, by mapping *i** to DMN. The proposed alignment is based on a semantic approach, by mapping the notation's elements to the concepts of a decision-making domain ontology.

In the past, we have investigated the alignment of *i** and BPMN [6, 12]. One could then argue for an approach combining all three dimensions: goal, business processes and decision modeling. Nevertheless, aligning DMN with *i** directly allows a more strategic (and thus, more abstract) view, instead of an operational one, which may prove useful in situations in which knowing in detail the activities leading to a decision is not necessary.

Main steps in our future work include the validation of the proposed alignment in general, both by conducting experiments and by applying it in different case studies. More generally, evolving the proposed ontology by adding other decision-making concepts will be performed iterating alignment refinement and validation activities.

References

1. van Heijst, G., van der Spek, R., Kruizinga, E.: The lessons learned cycle. In: Borghoff, U.M., Pareschi, R. (eds.) Information Technology for Knowledge Management, pp. 17–34. Springer, Heidelberg (1998). https://doi.org/10.1007/978-3-662-03723-2_2
2. OMG. Decision Model and Notation (DMN) v 1.1. (2016). https://www.omg.org/spec/DMN/About-DMN/. Accessed 28 Feb 2018
3. Janssens, L., Bazhenova, E., De Smedt, J., Vanthienen, J., Denecker, M.: Consistent integration of decision (DMN) and process (BPMN) models. In: España, S., Ivanović, M., Savić, M. (eds.) CEUR Workshop Proceedings CAiSE Forum 2016, pp. 121–128 (2016)
4. Giorgini, P., Maiden, N., Mylopoulos, J., Yu, E. (eds.): Social Modeling for Requirements Engineering. MIT Press, Cambridge (2010)
5. Guizzardi, R., Guizzardi, G.: Ontology-based transformation framework from tropos to AORML. In: [4], pp. 547–570 (2010)
6. Cardoso, E., Almeida, J.P., Guizzardi, R.: Analysing the relations between strategic and operational aspects of an enterprise: towards an ontology-based approach. Int. J. Organ. Des. Eng. **2**(3), 271–294 (2012)
7. Guizzardi, G.: Ontological foundations for structural conceptual models. Ph.D thesis. University of Twente, The Netherlands (2005)
8. Guizzardi, G., Falbo, R., Guizzardi, R.: Grounding software domain ontologies in the unified foundational ontology (UFO): the case of the ODE software process ontology In: IDEAS 2008, Recife, Brazil (2008)

9. van Lamsweerde, A.: Reasoning about alternative requirements options. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 380–397. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02463-4_20
10. Jiang, L., Barone, D., Amyot, D., Mylopoulos, J.: Strategic models for business intelligence. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 429–439. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24606-7_33
11. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 327–341. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41924-9_27
12. Marchetto, A., Nguyen, C.D., Di Francescomarino, C., Qureshi, N.A., Perini, A., Tonella, P.: A design methodology for real services. In: PESOS 2010, pp. 15–21 (2010)

Model-Driven Test Case Migration: The Test Case Reengineering Horseshoe Model

Ivan Jovanovikj[1(✉)], Gregor Engels[1], Anthony Anjorin[2], and Stefan Sauer[1]

[1] Software Innovation Lab, Paderborn University, Paderborn, Germany
{ivan.jovanovikj,engels,sauer}@upb.de
[2] Department of Computer Science, Paderborn University, Paderborn, Germany
anthony.anjorin@upb.de

Abstract. Existing test cases represent important assets, which are worth reusing in software migration projects. The benefit is twofold, reuse of relevant information as well cost saving by avoiding design of new test cases. As test cases are implemented in the same or a compatible technology as the system they are testing, they have to somehow follow the system migration, i.e., they should be co-migrated. Due to the size of the test case set, and often missing conformity in the structure of the test cases, migration of test cases is a quite challenging task. As model-driven engineering has been established to manage those complex tasks, we apply it in the test case domain. In this paper, we propose a generic migration method based on model-driven reengineering techniques. Our method which involves reverse engineering, restructuring, and forward engineering is applied in an industrial case study where appropriate tooling was developed as well.

Keywords: Software migration · Reengineering · Model-driven testing

1 Introduction

Software migration is a well-established method for transferring software systems into new environments without changing their functionality. For example, in legacy migration, a technological obsolete system that is still valuable for ongoing business is migrated into a new environment [10]. Another use case is when a system is migrated to another platform, thus enabling a multi-platform provisioning (a common use case in mobile application development nowadays).

Software testing plays an important role in software migration as it verifies whether the main requirement, i.e., the preservation of the functionality, is fulfilled. Existing test cases, are important assets and their reuse can be beneficial, not just from economical perspective, as the expensive and time consuming test design is avoided [40], but also from practical perspective: the existing test cases contain valuable information about the functionality of the source system. So similarly, as software migration is established to reuse existing systems, we want to reuse test cases as well.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 133–147, 2018.
https://doi.org/10.1007/978-3-319-92901-9_13

However, in many migration scenarios a direct reuse of test cases might be impossible due to system changes as well as the differences in the source and the target environments. As test cases are coupled with the system they are testing, the system changes need to be detected, understood and then reflected on the test cases to facilitate reuse. In other words, the test cases need to be co-migrated. But, the size of the test case set which sometimes is measured in tens of thousands of test cases, the missing conformity in the test case structure, make the migration of test cases a quite challenging task [28]. Hence, first of all, a new migration method should be able to deal with the structural complexity of the test cases. Then, it should also provide an easy way to restructure the test cases and reflect the relevant system changes. Last but not least, the migrated test cases should be appropriate for the target environment, i.e., the test cases have to be re-designed for the target environment.

The Model-Driven Engineering (MDE) software development methodology has been established to deal with the increasing complexity of the today's software systems by focusing on creating and exploiting domain models. The Model-Driven Architecture (MDA) initiative, proposed by Object Management Group (OMG), puts the MDE idea in action, and clearly separates the business logic from implementation by defining software models at different level of abstraction. The general idea is to distinguish between platform-independent models (PIMs) and platform-specific models (PSMs).

Software migration approaches that follow the MDA principles are known as model-driven software migration (MDSD) approaches [20]. Generally speaking, software migration can be seen as a transformation of the system which is done by enacting software transformation method, which is an instance of the well-known horseshoe model [29]. Therefrom, software migration is some kind of software reengineering, which according to [13] is defined as "examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form". In general, software reengineering consists of three consecutive phases: *Reverse Engineering*, *Restructuring* and *Forward Engineering*.

The MDA approach is also suitable for the test domain [22]. Software testing which relies on the MDA principles is known as model-driven testing [16,23,31]. Similarly to software migration, test case migration can be seen as a transformation of test cases implemented for a specific source technology to test cases implemented for a specific target technology.

Following the idea of model-driven software migration, we propose a reengineering horseshoe model for the domain of test cases. It contains the basic reengineering activities, *Reverse Engineering*, *Restructuring*, and *Forward Engineering* and artifacts on different levels of abstraction specific for software testing. Regarding the level of abstractions, it distinguishes between three levels of abstraction: *System Layer*, *Platform-Independent Layer*, and *Platform-Specific Layer*. Relying on the benefits from MDA and software reengineering, the main aim of the proposed reengineering horseshoe model is to provide (semi-)automated migration of test cases which ideally results in a high-quality migrated test case set, appropriate for the target environment. Only then one

can be sure that the execution of the migrated test cases is a clear indicator for the successfulness of the system migration.

In this paper, we also present a case study from an industrial project in which a migration of an existing modeling framework from Java to C# was performed. Using the activities and the artifacts proposed in the reengineering model, we firstly instantiated a model-driven test case migration method, then we developed appropriate tooling that was supporting the migration method, and at the end the method was enacted. In total, 4000 test cases were migrated from jUnit [3] to MSUnit [4], most of them completely automatic.

The structure of the rest of the paper is as follows: In Sect. 2, we give an overview of the fundamentals important for our work. Then, in Sect. 3, we present the test case reengineering horseshoe model. Section 4 discusses the case study where our reengineering model was applied. In Sect. 5 we discuss the related work and at the end, Sect. 6 concludes the work and gives an outlook on future work.

2 Background

Model-Driven Engineering (MDE) has been established to deal with the increasing complexity of development, maintenance and evolution of the nowadays software systems. In order to achieve this, it relies on models and model transformations [11]. Model-Driven Architecture (MDA), proposed by Object Management Group (OMG), defines several software models on different level of abstraction, thus clearly separating the business complexity from the implementation details [37]. MDA defines three levels of abstraction: *Computational-Independent Model (CIM)*, which focuses on the context and the requirements of the system, *Platform-Independent Model (PIM)*, which main focus is on the operational capabilities of the system without consideration of a specific technology and *Platform-Specific Model (PSM)*, which focuses on a specific platform.

Software migration is a well-established method for transferring software systems into new environments without changing their functionality. In case when MDA principles are followed, these migration approaches are known as model-driven software migration (MDSD) [20]. Software migration is some kind of software reengineering, which according to [13] is defined as "examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form". In general, software reengineering consists of three consecutive phases: *Reverse Engineering*, *Restructuring* and *Forward Engineering*. Reverse Engineering, according to [13], is the process of analyzing a subject system to create representations of the system in another form or on a higher level of abstraction. *Forward Engineering* is defined as "the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system" [13]. *Restructuring*, as defined by [13], is "the transformation from one representation form to another at the same relative abstraction level, while preserving the subject systems external behavior (functionality and semantics)".

Model-Based Testing (MBT) [38] is a software testing methodology that uses (semi-)formal models which encode the expected behavior of the system under test to derive test artifacts like test cases, test data or test configuration. Model-Driven Testing (MDT) [16,23] is a type of model-based testing which follows the Model Driven Engineering principles, i.e., the test cases are automatically generated from a test models using model transformations.

UML Testing Profile [36] is a language standardized by OMG which supports testing on model level. It can be divided into four main parts: *Test Architecture*, *Test Behavior*, *Test Data*, and *Test Management*. *Test Architecture* is used for specification of the structural aspects of the test environment and the corresponding test configuration. *Test Behavior* specifies the stimulation and observation of the SUT by using any kind of UML behavior diagram. Using *Test Data* one can specify the pre- and post-conditions as well as the input and expected output of the SUT. Last but not least, using *Test Management* one can manage for example, the test planing, scheduling or execution of test specifications.

Test Description Language (TDL) [18] is a testing language standardized by the European Telecommunications Standards Institute (ETSI)[1] bridges the gap between high-level test purpose specifications and executable test cases [41]. The language is scenario-based and it can be used for design, documentation, and representation of formalized test descriptions. The main ingredients of the language are *Test Data*, *Test Configuration*, *Test Behavior*, *Test Objectives* and *Time*. *Test Data* specifies the elements needed to express data sets and data instances which are used in test descriptions. Then, *Test Configuration* typed components and gates and the connections among the gates. *Test Behavior* defines the expected behavior. Using *Test Objectives* one can defines the objectives of the testing by attaching them to a behavior or to a whole test description.

Testing and Test Control Notation version 3 (TTCN-3) [17] is a testing language for test specification and implementation standardized by the European Telecommunications Standards Institute (ETSI). It supports creation of abstract test specifications which can be executed by providing additional implementation components, such as an SUT adapter.

xUnit [33] is a family of unit-testing frameworks that share common characteristics. Probably the most popular frameworks part of this family are jUnit [3] and nUnit [5] used for testing Java for C# software systems receptively. MSUnit is a Microsoft's testing framework for unit testing integrated in Visual Studio which has a similar, but still a little bit different structure compared to the xUnit-family frameworks.

3 The Test Case Reengineering Horseshoe Model

In this section, we present the test case reengineering horseshoe model [29] for migration of test cases which is defined by following the idea of model-driven software migration [19–21] (Fig. 1).

[1] http://www.etsi.org/.

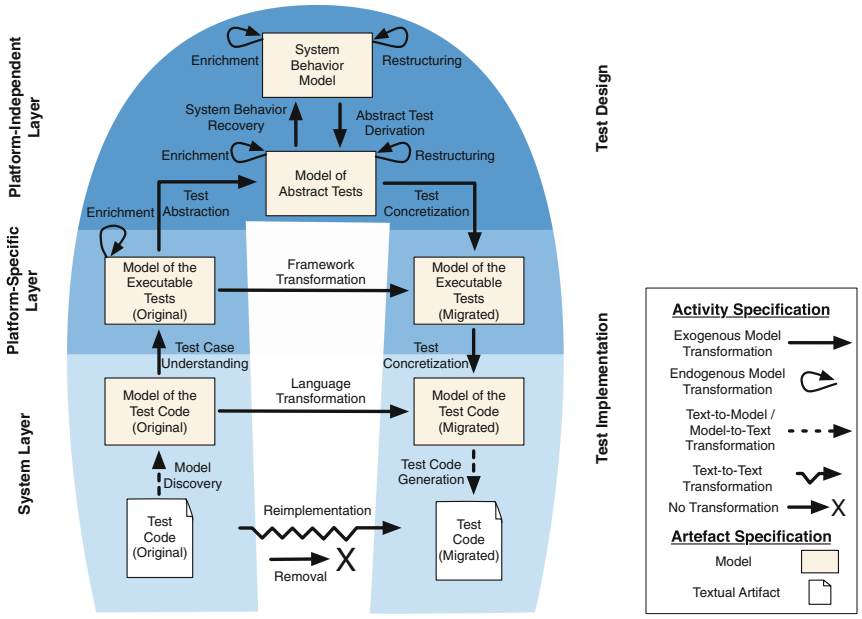


Fig. 1. The test case reengineering horseshoe model

It comprises the reengineering activities *Reverse engineering*, *Restructuring*, and *Forward Engineering* [13] and the corresponding artifacts in terms of textual artifacts and models on different levels of abstraction. Here, we present the conceptual method on model-driven migration of test cases and in the next section we come up with concrete examples from the case study that we have performed.

3.1 Artifacts

Artifacts are constituting parts of each transformation method. With regards to the level of abstraction, it can be distinguished between three layers of abstraction: *System Layer*, *Platform-Specific Layer*, and *Platform-Independent Level*.

System Layer
On the *System Layer*, textual artifacts representing test code and models of the test code are placed. Regarding the textual artifacts, this is either the original test code or the migrated test code. Similarly, regarding the models it is either the model of the original test code or the model of the migrated test code.

Test Code (Original/Migrated): The test code can be either the test cases, implemented in a specific testing framework, e.g. jUnit [3] or MSUnit [4], test configuration scripts or manually implemented additional test framework. If the

test cases are dependent on these artifacts, e.g., by using specific assert functions from the additional test framework, they also have to be considered and eventually migrated along with the test cases.

Model of the Test Code (Original/Target): The *Model of the Test Code* represents the test code in a form of Abstract Syntax Tree [9] of the appropriate language of the original or the target environment.

Platform-Specific Layer
The platform-specific layer is a higher level of abstraction compared to the system layer. Here, technology-specific concepts are used to represent the test cases for both the source and the target environment.

Model of Executable Tests (Original/Target): This model is considered as a platform-specific as it represents the executable test cases by using testing concepts which are specific for a particular testing framework, i.e., by using metamodels of jUnit [3] or MSUnit [4] to model the tests.

Platform-Independent Layer
On the *Platform-Independent Layer* the models representing the test cases are independent of any particular testing framework or testing technology. On this level of abstraction, we distinguish between two types of models, the *Model of the Abstract Tests* and the *System Behavior Model*.

Model of the Abstract Tests: This model is considered to be platform-independent as it is independent of any concrete testing frameworks. Standardized languages like UML Testing Profile (UTP) [36] and Test Description Language (TDL) [18] are used for modeling the abstract tests.

System Behavior Model: On the highest level of abstraction, we foresee the *System Behavior Model* that represents a compact representation of the expected behavior of the system. Behavioral diagrams like the UML activity or sequence diagram, or state machines can be used to represent the expected behavior of the system.

3.2 Activities

Activities in the test case reengineering horseshoe model produce or consume appropriate artifacts. In our model shown in Fig. 1, we distinguish between five types of activities: *Endogenous Model Transformation*, *Exogenous Model Transformation*, *Model-to-Text Transformation*, *Text-to-Model Transformation*, and *No Transformation* (removal/deletion of particular parts). These activities can be distinguished by the reengineering phase they belonging to. We rely on the classification defined by [13], where the reengineering is defined as process constituted of three phases: *Reverse Engineering*, *Restructuring*, and *Forward Engineering*.

Reverse Engineering. According to [13], *Reverse Engineering* is the process of analyzing a subject system to create representations of the system in another form or on a higher level of abstraction. When applied in the software testing domain, reverse engineering can be seen as a process of analyzing a test cases to create representations of the test cases in another form or on a higher level of abstraction, e.g., by using test models. In general, reverse engineering can be seen as combination of *Model Discovery* and *Model Understanding* [12].

The *Model Discovery* step relies on syntactical analysis and by using parsers in an automatic text-to-model transformation activity creates a model of the test case source code represented as an Abstract Syntax Tree (AST) [9]. The obtained model is known as initial model because it has a direct correspondence to the test cases. Model Discovery actually bridges the technical space of the test cases and the modelware technical space of MDE. The tools used to discover this models are known as model discoverers and for each specific language, an appropriate model discoverer is needed.

Model Understanding is a model-to-model transformation activity, or a chain of such activities, which takes the initial models, applies semantic mappings and generates derived models. This transformation performs a semantic mapping and results in a model of higher level of abstraction. In our reengineering model, we split the Model Understanding in three sub-activities: Firstly, the initial models are explored by navigating through their structure in an activity called *Test Case Understanding*. Model elements which represent test relevant concepts like test suite, test case or assertion are identified and then, by applying a model-to-model transformation, a test model of executable test cases is obtained. A model on this level is platform-specific which means that it is specific to a certain test technology and it is an instance of a metamodel of a particular testing framework (e.g., jUnit, MSUnit or TTCN-3). By applying the *Test Abstraction* activity, which is a model-to-model transformation as well, one can obtain a model of the abstract test cases which is platform-independent. As a platform-independent representation, UML Testing Profile (U2TP) [36] or Test Description Language (TDL) [18] may be used. The *System Behavior Recovery* activity is applied in order to obtain the highest possible level of abstraction defined by our reengineering model, the *System Behavior Model* which is a compact representation of the expected behavior of the system. Regarding the technical side of the model-to-model transformations, there are different transformation languages that can be used, e.g., QVT [35], JDT [1] or ATL [27].

Forward Engineering. As defined by [13], *Forward Engineering* is "the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system". In the field of software testing, this can be paraphrased as a process of moving of high-level test abstractions and logical implementation-independent design to the physical implementation of the test cases. The complete *Forward Engineering* side can be seen as Model-Based Testing, more specifically Model-Driven Testing [16,23]. The test models are used as input for a chain of

model-to-model transformations, ending with a model-to-text transformation, which provides the test code as output.

Abstract Tests Derivation takes as input the *System Behavior Model* and produces *Model of Abstract Test Cases*, a platform-independent model of the test cases. Then, by applying *Test Concretization* a *Model of Executable Test Cases* is obtained. For example, in case of UTP as model of the abstract tests and TTCN-3 as a model of executable tests, the mapping presented in [39] can be used as a base in the *Test Concretization* activity. This model is already specific for a particular testing framework and can be used for the generation of the *Test Code* by executing the *Test Code Generation* activity, which is a model-to-text transformation.

The tools which support forward engineering are known as generators and for each specific target platform, an appropriate generator is needed. Custom code generators can be built by using Xtend [8] which is a statically typed programming language which offers features like template expressions and intelligent space management.

Restructuring. According to [13], *Restructuring* is "the transformation from one representation form to another at the same relative abstraction level, while preserving the subject systems external behavior (functionality and semantics)". In the testing domain, we define test restructuring as the transformation from one representation to another at the same relative abstraction level, while preserving the "semantics" of the tests. With "semantics" we mean the functionality that is being checked by a particular test. This activity has been foreseen on both the *Test Model* as well as on the Model of *Abstract Test Cases*. The *Restructuring* activity is of course influenced by the target testing environment, testing tool, or by requirements on improving the quality of the test cases (e.g., maintainability). However, it could also be influenced by the changes that happen in the system migration. Since these changes may be relevant for the test models, they have to be reflected on the tests as well.

The *Reimplementation* is a Text-to-Text transformation which is performed by developers or testers manually, firstly observing the existing test cases an then implementing them for the target testing environment.

The *Language Transformation* also known as AST-based transformation [29], defines a direct mapping between the *Models of the Test Code*. This mapping between the original and the migrated model, is actually a mapping between the programming languages and the testing frameworks in which the test cases are implemented.

The *Framework Transformation* activity is performed on a higher level of abstraction and it defines a mapping directly between two testing frameworks, i.e., it defines a mapping between the testing concepts inside the original an the target testing framework.

The *Enrichment* activity is applicable to various models, e.g., Models of Executable Tests, Models of Abstract Tests or Test Models. By using annotation, one can insert an additional information to the tests.

The *Removal* activity is used to specify on which part of the test case code a transformation should not be performed. Due to the evolutionary development of the system as well as of the test cases, inconsistencies between the test code and the system code may exist, i.e., it may happen that no longer supported features or non-existing parts of the system are still being tested. Consequently, on those parts of the test code with obsolete test cases, a transformation is not performed. Intuitively, this activity compared to all other, is the only one that does not produce output.

Seen from the testing perspective, these three abstraction layers can be mapped to two testing abstraction layers. Namely, the *Platform-Independent Layer* is actually the *Test Design* Layer, where the initial test models are designed, the *System Behavior Model* of the SUT, and the *Model of the Abstract Tests*. Then, comes *Test Implementation*, where *Model of the Executable Tests* are derived which represent an implementation with concrete test data.

3.3 Tools

In order to support the previously introduced activities, thus enabling a (semi-) automatic transformation, we foresee in total three types of tools that are needed. Firstly, a parser is needed to obtain the initial models out of the textual artifacts, i.e, to perform the *Model Discovery* activity. Then, in a series of model-to-model transformations, which are specified by transformation rules, initial models are obtained. There rules are then executed by an Transformation Rule Engine. Finally, a *Test Code Generator* is needed to perform the model-to-text transformation by executing the test code generation rules previously specified, thus generating the test code for the migrated test cases.

4 Industrial Case Study

Our reengineering horseshoe model was applied in the context of an industrial project with the main goal of migrating parts of the well-known Eclipse Modeling Framework (EMF) [2] along with the Object Constraint Language (OCL) [6] from Java to C#. The old system is thus a Java program representing a subset of EMF and OCL, while the migrated system is an implementation of this subset as a corresponding C# program.

4.1 Migration Context

The primary change implemented by the system migration, besides switching from Java to C#, was to change from a Just-In-Time (JIT) compilation in the old system to an Ahead-Of-Time (AOT) compilation of OCL constraints in the migrated system.

As EMF and OCL are both well-tested frameworks, with all test cases being available on public code repositories [7], a major goal of the case study was to reuse the old OCL test cases to validate the migrated OCL functionality in the target environment.

It can be distinguished between two main types of source artifacts that have to be migrated, the test cases and the interlayer testing framework `TestOCL`. All an all, 13 different test suite are present, each of them addressing different functional aspects of OCL. On the first look, most of the test cases have similar structure. This characteristic is important, since we aim at automating the migration process of the test cases. The test cases that are following same structure are selected for automated migration. For those test cases that is difficult some repeating structure to identify, manual implementation is a more suitable solution. Additionally, an interlayer testing framework `TestOCL` exists in order to simplify the test cases. It provides domain-specific assert functions for the OCL domain, e.g., `assertQueryTrue`.

Regarding the language, the source system is implemented in Java under Java Platform. Regarding the architecture, OCL is implemented in that a native OCL expressions are specified and using the JAVA API they are evaluated. The language in the target environment is C# under the .net platform. The OCL functionalities are implemented directly in the corresponding classes that are specified in the migrated system. Regarding the transformation characteristics, the activities we ranging from manual to completely automatic. Those transformations that were performed automatically, have been formalized using transformation rules specified in Java. The process of transformation was supported by implementing transformation rules based on Xtend [8]. The thing of highest importance for the reflection of the changes to the test cases is the transformation rules specified for the transformation of OCL expression to C#. The change type can be seen as semantic-preserving, since the complete functionality of the OCL should be supported by the migrated system.

The source test environment is JUnit, with an additional interlayer extension, the *TestOCL*. The test cases are unit level test cases, separated in different test suites according the functionality they are addressing. The anatomy of the test cases in the target environment is that a single test case may contain multiple assertions, normally specified using the assert functions provided by the *TestOCL* extension. The complete change of the environment regarding language, framework, and architecture, implies a lot of changes in the testing context. The target environment language, in this particular case C#, implies usage of a C# Unit testing framework, MSUnit.

According to this context information, we came up with the migration method shown in Fig. 2. It is an instance of the previously introduced test case reengineering horseshoe (Fig. 1).

4.2 Implementation

To support the test case migration method, two Eclipse plug-ins [2] were developed, *TestCase2TestModel* and *TestModel2TestCase*. The names of the two plug-ins pretty much reveal what exactly the particular plug-in does. The *TestCase2TestModel* plug-in supports the activities on the left-hand side of the reengineering horseshoe, i.e., supports the reverse engineering activities, thus extracting test models from the test cases. The activities on the right-hand side

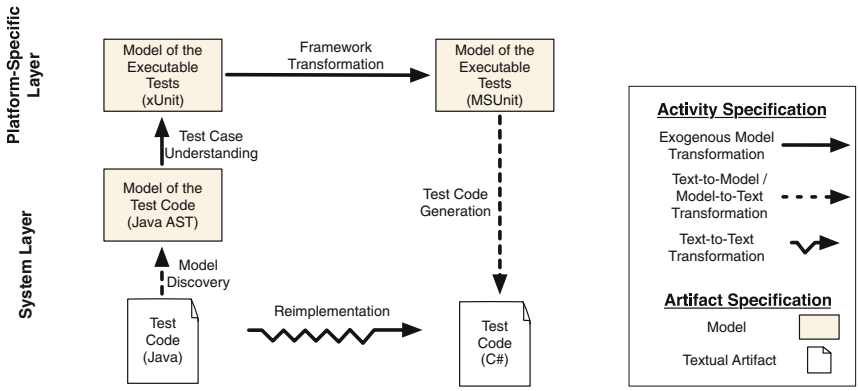


Fig. 2. The case study migration method: an instance of the test case reengineering horseshoe model

of the migration horseshoe, i.e., the forward engineering activities, are supported by the *TestModel2TestCase* plug-in which takes as input the test models and generates test code. Both plug-ins work in a completely automated way.

TestCase2TestModel. This plug-in supports the *Reverse Engineering* phase from the test case reengineering method. Consequently, it covers both the *Model Discovery* as well as *Test Case Understanding* activities. For *Model Discovery*, it uses the JDT Parser [1] to do a text-to-model transformation. It takes as input *jUnit* [3] test cases, i.e., a Java code and it produces an abstract syntax tree (AST). The AST is the input for the second activity, where out of the language-specific model elements, testing relevant concepts are identified and extracted. Out of this information, *Models of the Executable Test Cases* are produced which conform to the xUnit meta-model [33]. Thus, the newly obtained models contain model elements like test suite, test case or assertion, making the information explicit from testing point of view.

TestModel2TestCase. This plug-in covers the activities from *Restructuring* and *Forward Engineering* Phase. The transformation of OCL expressions from to C# has already been implemented. As the action in our test cases is actually to evaluate OCL expressions, we reuse the same transformation, to generate the test code in C#. The plug-in itself is a test code generator written using Xtend [8]. As input it takes the Xunit test models which represent the test suites and the test cases which are the outcome of the first plug-in. Using Xtend templates, test case code in MSUnit was generated for each corresponding test case from the Xunit test models. MS Unit Test Framework [4] was selected as a target testing framework environment for the test cases. As we have said, this plug-in reuses the same transformation from the system migration. Additionally, a special requirement regarding the anatomy of the test cases has to be followed,

i.e., the so called *"3-As" Pattern* had to be applied. The *"3-As" Pattern* defines that each test case on a code level is consisted of three parts: *Arrange*, *Act*, and *Assert*. For each of the three steps, OCL expressions have been translated to an appropriate C#-based form. By embedding the transformation initially created for the system migration, we were able to transform the test cases in the same way.

The overall result was an automated migration of over 92% of 4000 existing jUnit test cases. It is an open question if the migration of the test cases can further be automatized. 8% of the OCL test cases were not migrated automatically, because these are regression tests that have an irregular structure which complicates an automation.

5 Related Work

Regarding related work, we analyzed different research areas, software migration projects, and migration frameworks. Here, research areas like model-driven testing, test case reengineering, test mining and software migration are discussed. The area of model-driven testing includes already some proved methods that address the automation of the test case generation in a quite efficient way. In the work presented in [23], the authors aim to benefit from the separation of PIMs and PSMs in the generation and execution of tests by redefining the classical model-based tasks. Dai [14] proposes a model-driven testing methodology which takes as input UML system models and transforms them to test-specific models as instances of the UML Testing Profile. Javed et al. [26] propose a generic method for model-driven testing that relies on xUnit platform independent model. The work of Sousa et al. [26] represents an improvement of the work of Javed et al. in a way that they use a new test metamodel on the platform-independent level, whereas the xUnit metamodel is considered as a platform-specific. Lamancha et al. [30,31] propose also a model-driven testing method that relies on UML Testing Profile. Moreover, they present concrete transformations using the QVT transformation language. All of this methods are related to our work as they address the forward engineering side of our reengineering model. Regarding the reverse engineering side, we have identified also some existing work in the area known as test case reengineering. Hungar et al. [24] extract models out of test cases by means of automata learning. In [25], test models are synthesized from test cases by threating all test cases as a linear model and merging the corresponding states. The work of Werner et al. [42] goes in the similar direction and constructs trace graphs out of test cases to represent the system behavior. [15,34] go in the similar direction, and by exploiting the knowledge in the existing test cases written in Gherkin format, and extracting models in terms of finite state machines or state-flow graphs. Doing so, they aim to convert existing projects to model-based testing projects.

In a lot of software migration projects the importance of reusing test cases has been detected and an effort has been made to enable test case reuse. In the SOAMIG [32] migration project for example, existing test cases are used for the

analysis of the legacy system. MoDisco [12] is a generic and extensible framework devoted to Model-Driven Reverse Engineering. However, migration of test cases is still not addressed by this framework. The Artist [32] project proposes model-based modernization, by employing MDE techniques to automate the reverse engineering and forward engineering phases. From our point of view, it is the most interesting project as they also advocate migration of the test cases in a model-driven way, i.e., in a similar way the system has been migrated, thus reusing, above all, the developed tools. Our work differs in that way, that we propose a reengineering horseshoe model seen from software testing perspective.

6 Conclusion and Future Work

Inspired by the usage of Model-Driven Engineering and Model-Driven Architecture in software migration, in this paper we propose a test case reengineering horseshoe model which should be a fundamental step in enabling model-driven test case migration. It comprises the basic reengineering activities, *Reverse Engineering*, *Restructuring*, and *Forward Engineering* and artifacts on different levels of abstraction specific for software testing.

The proposed test case reengineering horseshoe model is a generic model with a high flexibility, as it can that can be used in different migration context for the migration of test cases. Employing a discipline like situational method engineering, one can come up with different test case transformation methods suitable for different migration contexts [21]. Therefore, we intend to use our reengineering model as a reference model in a method engineering approach, which would enable, according to a specific context, creation of an suitable migration method. Furthermore, our method relies on higher level of abstraction on well-established standards like the UML Testing Profile (UTP), an OMG standard, and on the especially interesting because of the growing popularity, Test Description Language (TDL), an ETSI standard. By developing appropriate tooling, also a high level of automation can be achieved thus reducing time and cost in the overall migration project. Last but not least, our method should enable co-migration of test cases in an migration scenario. Our main idea regarding this is to make the transformation activities of the reengineering model parameterizable, thus enabling reflection of the relevant transformation activities performed on the system.

References

1. Eclipse Java development tools (JDT). https://www.eclipse.org/jdt/
2. Eclipse Modeling Project. https://www.eclipse.org/modeling/emf/
3. JUnit. http://junit.org/junit4/
4. Microsoft Unit Test Framework. https://msdn.microsoft.com/en-us/library/hh598960.aspx
5. NUnit. http://nunit.org/
6. Object Constraint Language Specification Version 2.4. http://www.omg.org/spec/OCL/2.4/

7. Tests - org.eclipse.ocl.git - OCL. http://git.eclipse.org/c/ocl/org.eclipse.ocl.git/tree/tests/
8. Xtend - Modernized Java. http://www.eclipse.org/xtend/
9. Architecture-driven Modernization: Abstract Syntax Tree Metamodel (ASTM)-Version 1.0. Object Management Group (2011). http://www.omg.org/spec/ASTM/1.0/PDF/
10. Bisbal, J., Lawless, D., Wu, B., Grimson, J.: Legacy information systems: issues and directions. IEEE Softw. **16**(5), 103–111 (1999)
11. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, 1st edn. Morgan & Claypool Publishers, San Rafael (2012)
12. Bruneliere, H., Cabot, J., Jouault, F., Madiot, F.: MoDisco. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering - ASE 2010, p. 173. ACM Press, New York (2010)
13. Chikofsky, E.J., Cross, J.H.: Reverse engineering and design recovery: a taxonomy. IEEE Softw. **7**(1), 13–17 (1990)
14. Dai, Z.R.: Model-Driven Testing with UML 2.0. Computing Laboratory, University of Kent (2004)
15. Dixit, R., Lutteroth, C., Weber, G.: FormTester: effective integration of model-based and manually specified test cases. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, pp. 745–748. IEEE (2015)
16. Engels, G., Güldali, B., Lohmann, M.: Towards model-driven unit testing. In: Kühne, T. (ed.) MODELS 2006. LNCS, vol. 4364, pp. 182–192. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69489-2_23
17. E.T.S.I.: ETSI Standard ES 201 873–1: The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language, V3.1.1 (2005)
18. E.T.S.I.: ETSI ES 203 1191: Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 1: Abstract Syntax and Associated Semantics, v1.3.1 (2016)
19. Fleurey, F., Breton, E., Baudry, B., Nicolas, A., Jézéquel, J.-M.: Model-driven engineering for software migration in a large industrial context. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 482–497. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75209-7_33
20. Fuhr, A., Winter, A., Erdmenger, U., Horn, T., Kaiser, U., Riediger, V., Teppe, W.: Model-driven software migration - process model, tool support and application. In: Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments, pp. 153–184 (2012)
21. Grieger, M.: Model-driven software modernization: concept-based engineering of situation-specific methods. Ph.D. thesis, University of Paderborn, Germany (2016)
22. Gross, H.G.: Testing and the UML: a perfect fit. Technical report 110, Kaiserslautern (2003)
23. Heckel, R., Lohmann, M.: Towards model-driven testing. In: Electronic Notes in Theoretical Computer Science. vol. 82, pp. 37–47. Elsevier (2003)
24. Hungar, H., Margaria, T., Steffen, B.: Test-based model generation for legacy systems. IEEE International Test Conference (ITC), Charlotte, NC, 30 September - 2 October 2003 (2003)
25. Jääskeläinen, A., Kervinen, A., Katara, M., Valmari, A., Virtanen, H.: Synthesizing test models from test cases. In: 4th International Haifa Verification Conference on Hardware and Software: Verification and Testing, pp. 179–193 (2009)

26. Javed, A.Z., Strooper, P.A., Watson, G.N.: Automated generation of test cases using model-driven architecture. In: Second International Workshop on Automation of Software Test (AST 2007), p. 3. IEEE (2007)
27. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008)
28. Jovanovikj, I., Grieger, M., Yigitbas, E.: Towards a model-driven method for reusing test cases in software migration projects. In: Softwaretechnik-Trends, Proceedings of the 18th Workshop Software-Reengineering & Evolution (WSRE) & 7th Workshop Design for Future (DFF), vol. 32, no. (2), pp. 65–66 (2016)
29. Kazman, R., Woods, S., Carriere, S.: Requirements for integrating software architecture and reengineering models: CORUM II. In: Proceedings Fifth Working Conference on Reverse Engineering, pp. 154–163. IEEE Computer Society (1998)
30. Lamancha, B.P., et al.: Automated model-based testing using the UML testing profile and QVT. In: Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation, pp. 1–10 (2009)
31. Lamancha, B., Reales, P., Polo, M., Caivano, D.: Model-driven test code generation. Commun. Comput. Inf. Sci. **275**, 155–168 (2013)
32. Menychtas, A., et al.: Software modernization and cloudification using the ARTIST migration methodology and framework. Scalable Comput.: Pract. Exp. **15**(2), 131–152 (2014)
33. Meszaros, G.: XUnit Test Patterns: Refactoring Test Code. Addison-Wesley, Boston (2007)
34. Milani Fard, A., Mirzaaghaei, M., Mesbah, A.: Leveraging existing tests in automated test generation for web applications. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, pp. 67–78 (2014)
35. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Version 1.1 (2011). http://www.omg.org/spec/QVT/1.1/
36. OMG: UML Testing Profile (UTP), Version 1.2. (2013). http://www.omg.org/spec/UTP/1.2/
37. OMG: Model Driven Architecture (MDA): MDA Guide rev.2.0 (2014). http://www.omg.org/cgi-bin/doc?ormsc/14-06-01
38. Pretschner, A., Philipps, J.: 10 methodological issues in model-based testing. In: Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., Pretschner, A. (eds.) Model-Based Testing of Reactive Systems. LNCS, vol. 3472, pp. 281–291. Springer, Heidelberg (2005). https://doi.org/10.1007/11498490_13
39. Schieferdecker, I., Dai, Z.R., Grabowski, J., Rennoch, A.: The UML 2.0 testing profile and its relation to TTCN-3. In: Hogrefe, D., Wiles, A. (eds.) TestCom 2003. LNCS, vol. 2644, pp. 79–94. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44830-6_7
40. Sneed, H.: Risks involved in reengineering projects. In: Sixth Working Conference on Reverse Engineering, pp. 204–211 (1999)
41. Ulrich, A., Jell, S., Votintseva, A., Kull, A.: The ETSI test description language TDL and its application. In: 2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 601–608 (2014)
42. Werner, E., Grabowski, J.: Model reconstruction: mining test cases. In: Third International Conference on Advances in System Testing and Validation Lifecycle. VALID 2011 (2011)

MICROLYZE: A Framework for Recovering the Software Architecture in Microservice-Based Environments

Martin Kleehaus(✉), Ömer Uludağ(✉), Patrick Schäfer(✉), and Florian Matthes(✉)

Chair for Informatics 19, Technische Universität München (TUM), 85748 Garching, Germany
{martin.kleehaus,oemer.uludag,patrick.schaefer,matthes}@tum.de

Abstract. Microservices are an approach to distributed systems that promote the use of finely grained services with their own lifecycles. This architecture style encourages high decoupling, independent deployment, operation and maintenance. However, those benefits also leave a certain aftertaste, especially in continuous documentation of the overall architecture. It is fundamental to keep track of how microservices emerge over time. This knowledge is documented manually in Enterprise Architecture (EA) tools, which leads to an obsolete status. For that reason, we present a novel multi-layer microservice architecture recovery approach called *MICROLYZE* that recovers the infrastructure in realtime based on the EA model involving the business, application, hardware layer and the corresponding relationship between each other. It leverages existing monitoring tools and combines the run-time data with static built-time information. Hereby, *MICROLYZE* provide tool support for mapping the business activities with technical transactions in order to recover the correlation between the business and application layer.

1 Introduction

The popularity of microservice-based architectures [11] is increasing in many organizations, as this new software architecture style introduces high agility, resilience, scalability, maintainability, separation of concerns, and ease of deployment and operation [4]. Adrian Cockcroft at Netflix describes the architecture style as a "fine grained SOA" [7] that presents single applications as a suite of small services that run in their own processes and communicate with each other through lightweight HTTP-based mechanisms, like REST. Microservices are built around business capabilities and enclose specific business functions that are developed by independent teams [10]. The benefits emphasize the reason why over the last decade, leading software development and consultancy companies have found this software architecture to be an appealing approach that leads to more productive teams in general and to more successful software products. Companies such as Netflix [30], SoundCloud [6], Amazon [22] or LinkedIn [20] have adopted this architecture style and pioneered the research in this area.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 148–162, 2018.
https://doi.org/10.1007/978-3-319-92901-9_14

Even though microservices release the rigid structure of monolithic systems due to the independent deployment, this style also introduces a high level of complexity with regard to architecture monitoring, recovery and documentation [4]. In a recently published study on architecting microservices [12], it was investigated that monitoring solutions for microservice-based architectures have been addressed by many researchers and companies. However, it was also confirmed that limited research was conducted on topics of microservice recovery and documentation, although this is very important for understanding the emerging behavior of microservice-based architectures. For instance, microservices can dynamically change their status at run-time like the IP address or port for multiple reasons like autoscaling, failures, upgrades, or load balancing, among others [13]. Additional services can be introduced into the infrastructure or removed during upgrades or migration projects. In these scenarios, it is crucial to keep track on the current architecture orchestration and service dependencies.

In order to tackle this challenge, a lot of different monitoring approaches [2,3,21,28] and service discovery mechanisms [23,24] are applied that pull the status of services dynamically over the network. This simplifies many aspects of tracking the system, but lacks in connecting their obtained monitoring data in order to achieve an integrated and holistic view of the behavior and status of the whole Enterprise Architecture (EA) [5]. Moreover, the alignment of business needs to the IT infrastructure is acquiring increasing importance in microservice environments. Best practices propose the design of services based on their business domain in which they fulfill one specific business requirement [10]. Hence, we are convinced that the documentation of the relationship between the services and business domains including departments, teams and business processes has to be involved in an holistic architecture recovery, which is not yet covered in existing approaches.

According to the aforementioned considerations, we propose an architecture recovery approach called *MICROLYZE*, which combines static and runtime data in order to reconstruct IT infrastructures that are based on microservice architecture. The reconstruction includes all adjacent layers proposed by EA models, like hardware, application, business layer, and their relationships between each other. The tool recognizes changes in the infrastructure in real-time and uncovers all business activities that are performed by the users. The prototype obtains most of the required data via well-selected monitoring tools. The presented approach was evaluated in a microservice-based system called TUM Living Lab Connected Mobility (TUM LLCM)[1].

The rest of the paper is organized as follows: Sect. 2 describes the layers and components that are involved in the architecture recovery. Section 3 presents the technical details of the architecture recovery approach. In Sect. 4 we evaluate our approach in a real case scenario, whereas Sect. 5 provides a benchmark about the instrumentation overhead. Sections 6 and 7 closes the paper with related work and our future efforts to improve the proposed approach.

[1] http://tum-llcm.de/.

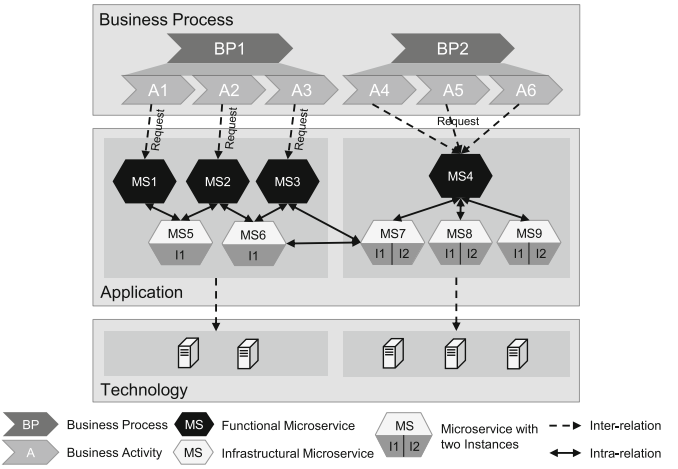


Fig. 1. Multi-layered microservice-based EA infrastructure

2 Architecture Model

MICROLYZE aligns the reconstruction model of microservice-based infrastructures to the EA model adopted by many of the EA frameworks that emerged in the last decades, like ArchiMate [15], Zachman [31], TOGAF [17], amongst others. These frameworks provide standards on how to model the EA and typically divide it into three abstraction layers: (1) the technology layer encompasses all technological-related aspects like hardware, network and other physical components and (2) the application layer defines software components running on the technology layer. We assign services and service instances to the application layer. (3) The business layer operates on top of the aforementioned layers and defines all business-related aspects like the business departments, business processes and business activities that are processed by the microservices. The first two layers are reconstructed completely automatically via analyzing monitoring data; the business layer requires additional domain knowledge and manual input in the first place that can only be provided by department staff members. The overall architecture and the relationship between each layer is depicted in Fig. 1 and described in more detail in the following.

2.1 Business Process

A business process is a collection of related activities that serve a particular goal for a user. In the context of distributed systems, each business transaction, or technically speaking a user request, contributes to the execution of a business process. Hence, several organizations and the according teams who develop the services could be involved in one business process. The reconstruction of a business process with the aid of event data is mostly associated with process discovery, which is the most used technique in process mining [1].

2.2 Business Activity

A business activity defines a business transaction and consists of a sequence of related events that together contribute to serve a user request. A request represents interactions with the system. Every business transaction is part of one or more business processes. Transactions can span multiple microservices that expose external interfaces to other services for intercommunication. Before process mining can be accomplished, each technical request that represents a business-related user activity must be named with a clear and understandable business description.

2.3 Service

A service is a logical unit that represents a particular microservice application. According to the service type classification discussed by Richards [25], services can be classified into functional and infrastructural services. Infrastructure services are not exposed to the outside world but are treated as private shared services only available internally to other services. Functional services are accessed externally and are generally not shared with any other service. They are responsible for processing business transactions but can forward the user request to other infrastructure services.

2.4 Service Instance

In contrast to services that form the logical unit of a microservice, the service instance represents the real object of this service. We introduce this separation as a logical service can *own* more service instances. This is often the case when load balancing is applied. However, a service is always represented by at least one instance. The instances are identified by the used IP address and port but always contain the same endpoint service name.

2.5 Hardware

The hardware layer covers all physical components of a microservice infrastructure. The service instances run on the hardware. We assume that hardware can be identified by its IP address.

2.6 Relationship Between Architecture Components

The architecture model depicted in Fig. 1 constitutes two relationship types between the components: The intra-relation defines connections within a specific abstraction layer. For instance, as mentioned above, a business process is a sequence of business transactions and every transaction is defined by the organization that managed the service. In the application layer as an example, several services contribute to serve a user request. These services exchange data over their interface and, hence, feature an intra-relationship.

Besides relationships within abstraction layers, the inter-relation constitutes connections between two different layers. In order to obtain the holistic architecture of a microservice-based environment, inter-relationships uncover important information about the interaction between abstraction layers. All functional services are deployed to process specific business transactions that are defined by the executed user request. By tracing these requests, application-related metrics can be obtained, like the duration of a user request, the latency between two distributed services or the data volume that is transferred. Due to the inter-relationships, system administrators are able to quickly identify which business activities or business processes are affected by this failure. In addition, one can point out the team who is responsible for fixing the error.

3 Recovery Process

We regard architecture recovery as a never-ending process. It is a continuous monitoring of the service interaction within an IT infrastructure. Microservice architectures evolve over time [9]. That means, new services are added or removed, and newly implemented interfaces lead to a change in the information exchange and dependency structure. For that reason, in order to find a proper monitoring approach it is a prerequisite to receive data about the current health status of the application and insights about the communication behavior between microservices. It is important to keep track of architectural changes, especially when new releases cause failures or performance anomalies. In addition, the architecture recovery process should not only cover technology-related aspects but also business-related points, since each architecture refinement could add additional endpoints for interacting with the application. This leads, in turn, to an extension of the business process. Therefore, new business activities must be recognized automatically and added to the responsible business process.

Based on the aforementioned considerations, our architecture recovery process is composed of six phases as illustrated in Fig. 2. In the **first phase**, *MICROLYZE* automatically rebuilds the current microservice infrastructure that is registered in a *service discovery* tool like Eureka[2] or Consul[3]. These systems are integrated in microservice-based environments for storing the instance information of running microservices. Microservices frequently change their status due to reasons like updates, autoscaling or failures; the service discovery mechanisms are used to allow services to find each other in the network dynamically. By retrieving the information from the *service discovery* service, we are able to reveal the current status of each service instance. In case a change (unregistered service, new service, updated service) is detected *MICROLYZE* alters the discovered architecture structure in order to indicate this particular change.

In the **second phase**, we also use the retrieved information for recovering hardware related aspects in order to establish the link between the microservices

[2] https://github.com/Netflix/eureka.
[3] https://www.consul.io/.

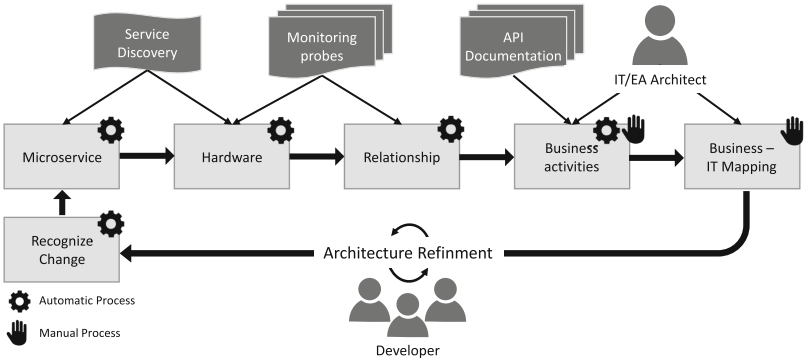


Fig. 2. Microservice architecture discovery process with all required phases (1–6) and included data sources

and the hardware on which the services are running. The *service discovery* service already provides useful data like, IP address and port but lacks in reporting detailed hardware information. For that reason, installing an additional monitoring agent on each hardware component that reveals hardware related information is required. The IP address is used to establish the link between the application and hardware layer.

Although service discovery mechanisms are often applied to discover the status of running services in run-time, they mask the real dependencies among microservices in the system. It remains unknown how the services communicate with each other as soon as user transactions come in. For that reason, it is necessary to install on each microservice a monitoring probe that supports the distributed tracing technology introduced by Google [29]. Distributed tracing tracks all executed HTTP requests in each service by injecting tracing information into the request headers. Hereby, it helps to gather timing data like process duration for each request in order to troubleshoot latency problems. Furthermore, additional infrastructure and software-specific data like endpoint name, class, method, HTTP request, etc. is collected and attached as annotations. Distributed tracing uncovers the dependencies between microservices by tracking the service calls via a correlation identifier. It is also capable of differentiating between concurrent or synchronous calls. As an implementation of the distributed tracing technology, we refer to the open-source monitoring solution zipkin[4] developed by Twitter. It supports the openTracing standard [5] and enjoys a huge community. Hence, by means of zipkin we are able to talk with any other APM tool as long as it also supports the openTracing standard. The zipkin probes stream the information via apache kafka to *MICROLYZE* and are stored in a cassandra database.

[4] https://github.com/openzipkin/zipkin.
[5] http://opentracing.io/.

Moreover, we implemented an algorithm that determines how to classify each service on basis of the distributed tracing data. *Functional services*, for instance, have mostly no parent services that forward the request to their child nodes. The very first application is the client itself. Hence, the parent ID in the tracing data is mostly empty. However, there are situations in which this approach is not applicable. *Gateway services*, for example, provide a unified interface to the consumers of the system that proxies requests to multiple backing services. In order to recognize this type of service, we continuously analyze the incoming HTTP requests. If the very first accessed microservice is always the same in most requests *MICROLYZE* flags it as the *gateway service*. All child nodes after the gateway are flagged as *functional services* accordingly.

Last but not least, huge microservice infrastructures are load balanced to avoid single points of failures. Instances of a service always have the same name but distinguish itself in IP address and port. Therefore, the uniqueness of a service instance is defined by the service description in combination with the used IP address and the service port. In order to discover all instances that belong to a specific service, we aggregate them based on the service description.

During the **third phase**, all transactions executed by users are stored in a database. These requests provide information about the user behavior, which includes, first of all, what the user does, when and in which chronological order, but also uncovers the link between the business transactions and the microservices that are responsible for processing the request. However, most monitoring or CMDB solutions do not establish a mapping between the business activities and the technical transactions. It remains unclear which service is responsible for processing a specific business activity. For that reason, we extend *MICROLYZE* with a *business process modeller* that assists in creating such a mapping.

First of all, in the **fourth phase** each business activity that can be performed by the users and triggers a request in the backend are defined with a clear semantic description like *register, open shopping cart, purchase article*, etc. After this step is finished, we are able to create the mapping between the business activities and the technical requests extracted by zipkin. This covers **phase five**. In order to support this process, we enhanced the *business process modeller* with the regular expression language in order to describe technical requests. These expressions are mapped with the previously described business activities. All expressions are stored in the database and validate incoming requests. Hence, new incoming transactions that are not yet seen and might refer to a modelled business activity are already mapped by a regular expression.

The **sixth phase** is all about recognizing changes in the IT infrastructure. The user is notified as soon as the system recognizes changes in the architecture model that might occur after a component update. *MICROLYZE* frequently polls the *service discovery* service that indicates deleted or newly added services that are not known yet. Unregistered services could indicate a service crash or a new update release. For that reason, they are not automatically removed from the database but only marked as such. This flag is removed as soon as the services are registered again. Unknown user requests that cannot be validated by a predefined

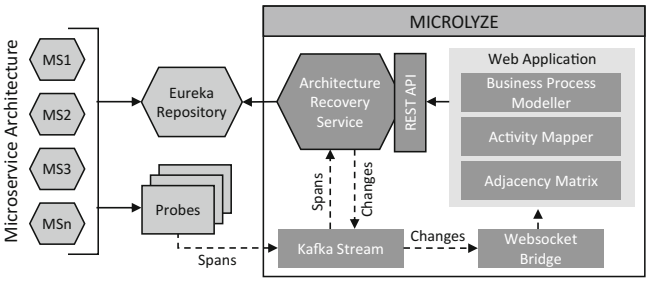


Fig. 3. Microlyze architecture components

regular expression are added to the list of unmapped URL endpoints. These endpoints have to be linked to a business activity afterwards. Changes in IP addresses and port might indicate an alteration in the underlying infrastructure, which leads to an automatic adaption of the architecture model.

The overall microservice architecture is depicted in Fig. 3. Apache kafka is used to stream all monitoring spans and architecture changes to the prototype in realtime. The web application includes the business process modeller, the activity to service mapper and the architecture visualizer that renders the obtained architecture model, making the discovered information, components, dependencies and mappings during the recovery process available for system administrators and enterprise or software architects. As microservice environments can grow to hundreds of services, we chose a representation that is scalable as well as capable of handling hundreds of dependencies and hiding specific aspects that are currently not needed to visualize. For that purpose, we applied the concept of the adjacency matrix and grouped the related columns and rows to the component dimensions that were described in Sect. 2.

4 Evaluation

The described architecture discovery concept has been prototyped and applied to the TUM LLCM platform. This platform simplifies and accelerates the exchange regarding the development of digital mobility services. It allows service developers to build better mobility services by incorporating different partners who agree to share their mobility data. In this context, a service called *Travelcompanion* was developed that enables travelers to connect with a travel group who has the same destination. Hereby, travel costs can be shared between the group members. The *Travelcompanion* service consumes data from a BMW DriveNow and Deutsche Bahn (DB) service and provides a recommendation on how to plan the route with different transportation means used by other travel groups that still have free space left. By joining these groups, the travel costs can be minimized.

The platform is a microservice-based system implemented in Spring Boot. Each service runs in a docker container. The architecture incorporates infrastruc-

tural services like a configuration (config-service), the service discovery eureka (eureka-service) and a gateway service (zuul-service). Further services provide administration (business-core-service), geospatial (maps-helper-service) and accounting (accounting-core-service) functionality like booking and payment. DriveNow (drivenow-mobility-service), DB (deutschebahn-mobility-service) and the Travelcompanion (travelcompanion-mobility-service) service have their own data storage. Each service represents a eureka client and is registered in the eureka server. Each service except eureka is instrumented by zipkin probes. The microservice architecture is distributed on three virtual machines, each running on the same hardware.

After each service is started, the architecture discovery accesses eureka and consumes all registered services. As the matrix in Fig. 4 shows, *MICROLYZE* correctly recognizes 9 services (S1–S9), 9 instances (I1–I9) and 3 hardware components (H1–H3). Each service is assigned to only one instance, which uncovers there is no load balancing in place.

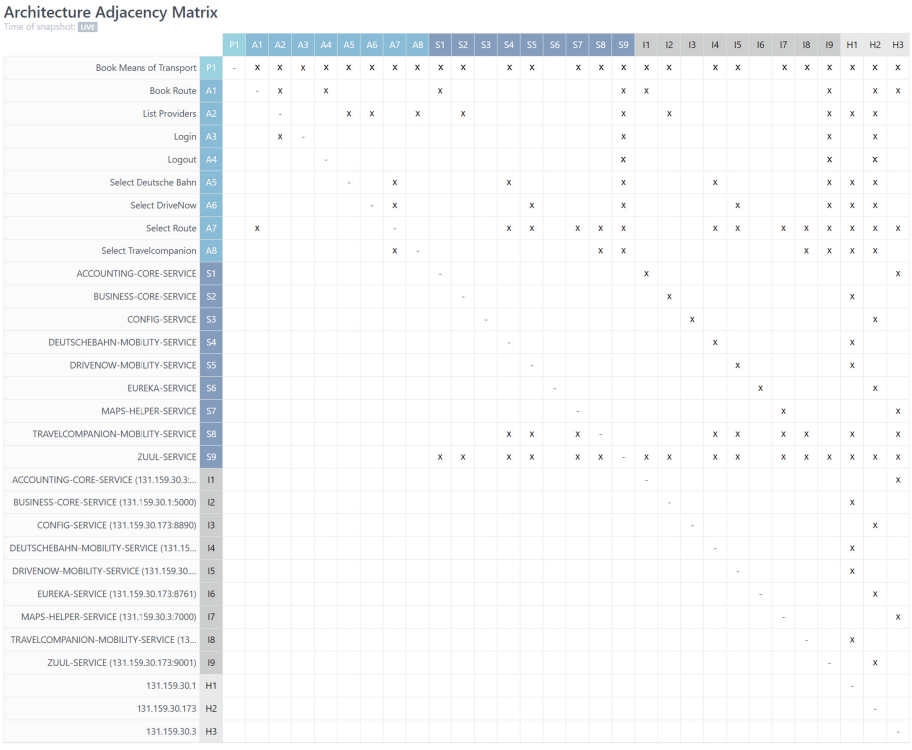


Fig. 4. Architecture discovery result visualized in a grouped adjacency matrix

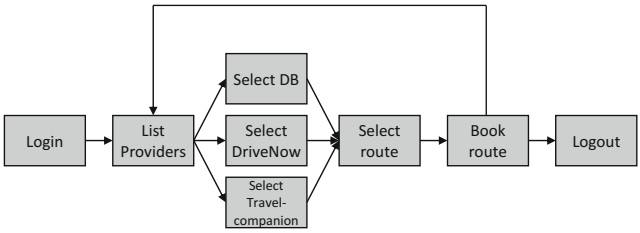


Fig. 5. Simulated user journey through the TUMLLCM platform. The business process describes the booking of a transportation mean

In order to recover the relationships between the services, we produce traffic on the platform by using JMeter[6], which simulates user transactions based on the given REST API endpoints documented by Swagger. After each endpoint was called *MICROLYZE* is able to reconstruct the dependency structure among the microservice architecture. The adjacency matrix visualizes that service S8 (travelcompanion-mobility-service) consumes data from service S4, S5 and S7, which is intended. The communication between S8 → S4 and S8 → S5 is asynchronously which is also correct. In addition, it is detected that service S9 consumes data from every non-infrastructural service. Hence, *MICROLYZE* successfully recognizes S9 as the gateway service.

As soon as step three of the discovery process is finished and the system has collected enough transaction data, we start to enhance the technical transactions with a business semantic. The user click journey, which we want to simulate, is depicted in Fig. 5. Although this process is rather small it could grow to hundreds of user clicks that had to be modelled in *MICROLYZE*. In order to accelerate this step or even circumvent it, we could apply a business process mining approach and import the XML model into *MICROLYZE*. In the scope of this work, we modeled the process manually. For simplicity, each business activity ranks among the same business process "Book Means of Transport". In total, we create one process containing eight activities (A1–A8) as shown in the adjacency matrix. Afterwards, we proceed with the mapping of each business activity to a user transaction. Hereby, *MICROLYZE* supports this step by providing a framework for regular expressions that describe a particular transaction. For instance, the expression */[0–9]+/book$* is mapped to the activity "book route". In order to define the business instance for recovering the performed business process, we apply the user session as our business case.

The matrix in Fig. 4 visualizes the final result. The execution of business process P1 involves every service except the infrastructural services S3 and S6. Each activity is processed by service S9, which provides further proof that S9 represents a gateway service. Moreover, it is clearly visible that for performing activity A7 also service S7 is executed besides the mobility services S4, S5 and S8.

[6] http://jmeter.apache.org/.

This is due to the communication dependency between service S8 and S7. Hence, service S7 is indirectly involved with performing A7.

By hovering over a dependency field within the matrix, an information box is displayed and shows relation specific information. This information includes all relation annotations, for instance, the communication between S8 → S4 and S8 → S5 is asynchronously which is correctly reported.

5 Instrumentation Overhead

We investigate the extent of instrumentation overhead via a performance benchmark. We measure the time to complete each business activity for both the instrumented and unmodified version of the software. Each activity executes a transaction that is initially processed by the gateway service. The following <business activity>:<service> pairs are involved:

- List Providers: Business service
- Select Travelcompanion: Travelcompanion service
- Select Route: Map, DriveNow, DB service
- Book route: Accounting service.

This time measurement is performed on the user side, and thus includes the communication overhead between the application and the client user. By measuring on the client side, we achieve an end-to-end processing benchmark. We repeated these measurements several times and calculated the average run-time and associated a 95% confidence interval. The results are presented in Fig. 6. We use JMeter to perform each request 5000 times involving database querying. As Fig. 6 illustrates, the difference in performance is very small. On average, the requests take 2 ms longer to respond. Based on the observations presented above, we conclude that the impact of the instrumentation is negligible.

6 Related Work

O'Brien et al. [27] provide a state-of-the-art report on several architecture recovery techniques and tools. The presented approaches aim to reconstruct software components and their interrelations by analyzing source code and by applying data mining methods.

O'Brien and Stoermer [26] present the Architecture Reconstruction and MINing (ARMIN) tool for reconstructing deployment architectures from the source code and documentation. The proposed reconstruction process consists of two steps: extracting source information and architectural view composition. In the first step, a set of elements and relations is extracted from the system and loaded into ARMIN. In the second step, views of the system architecture are generated by abstracting the source information through aggregation and manipulation. ARMIN differs from our approach, as it only extracts static information of the system without considering dynamic information.

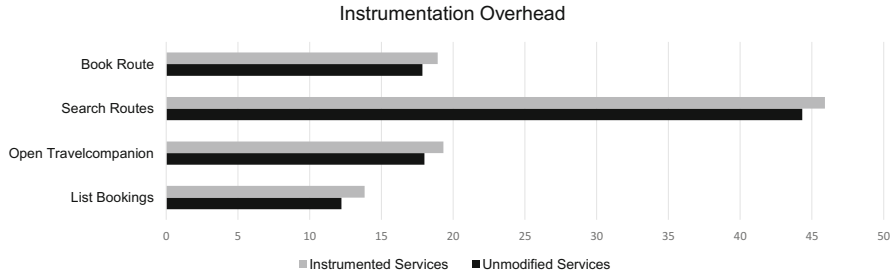


Fig. 6. Effect of instrumentation on the average time to complete – average time to complete (in milliseconds) [95% confidence interval]

Cuadrado et al. [8] describe a case study of the evolution of an existing legacy system towards a SOA. The proposed process comprises architecture recovery, evolution planning, and evolution execution activities. Similar to our approach, the system architecture is recovered by extracting static and dynamic information from system documentation, source code, and the profiling tool. This approach, however, does not analyze communication dependencies between services, which is an outstanding feature of our prototype.

van Hoorn et al. [18,19] propose the java-based and open-source Kieker framework for monitoring and analyzing the run-time behavior of concurrent or distributed software systems. Focusing on application-level monitoring, Kieker's application areas include performance evaluation, self-adaptation control, and software reverse engineering, to name a few. Similar to our approach, Kieker is also based on the distributed tracing for uncovering dependencies between microservices. Unlike us, Kieker does not process architectural changes in runtime. Furthermore, it does not cover dependencies between the business and application layer.

Haitzer and Zdun [16] present an approach for supporting semi-automated abstraction of architectural models supported through a domain-specific language. The proposed approach mainly focuses on architectural abstractions from the source code in a changing environment while still supporting traceability. Additionally, the approach allows software architects to compare different versions of the generated UML model with each other. It bridges the gap between the design and the implementation of a software system. In contrast, we propose an approach for recovering microservices in order to overcome the documentation and maintenance problem.

MicroART, an approach for recovering the architecture of microservice-based systems is presented in [13,14]. The approach is based on Model-Driven Engineering (MDE) principles and is composed of two main steps: recovering the deployment architecture of the system and semi-automatically refining the obtained system. The architecture recovery phase involves all activities necessary to extract an architecture model of the microservices, by finding static and dynamic information of microservices and their interrelations from the

GitHub source code repository, Docker container engine, Vagrant platform, and TcpDump monitoring tool. The architecture refinement phase deals with semi-automatically refining the initial architecture model by the architect. MicroART considers the architecture model of the microservices without regarding the business layer, which is a main feature in our approach. Furthermore, MicroART does not differentiate types of microservices interrelations, like synchronous or asynchronous. Moreover, it does not rebuild the architecture model as soon as architectural changes emerge.

7 Conclusion

In this paper, we presented a novel approach to recover the architecture from microservice-based systems. The discovery solution is based on a layered structure proposed by recommended EA frameworks. *MICROLYZE* is capable of recreating the dependencies between the business and application layer by providing a smart business-it mapper, and the hardware layer. The recovery process itself is subdivided into six phases. In the first three phases, we combine the monitoring data from a service discovery repository and a distributed tracing solution in order to reconstruct the microservice architecture, the dependencies between each service and the underlying hardware infrastructure. In the next phases, we describe each technical request as a business activity and map these activities to a business process. The concept and the tool have been successfully applied to the TUMLLCM microservice platform, which was able to recover the infrastructure without any error.

The proposed approach works well if two implementations are presented. First of all, each service has to be instrumented by an application performance monitoring solution that supports distributed tracing and complies with the open tracing standard. Furthermore, a *service discovery* service like Eureka or Consul has to be integrated. In case one of those tools is not installed, *MICROLYZE* will not become fully operational, which presents our most significant limitation.

In our future work, we plan on storing each architecture change in order to travel through the architecture evolution and on comparing the past and the current status. This feature will uncover important insights about the emerging behavior of microservice architectures, especially after new releases. It allows, for example, the analysis of the as-is and the intended to-be status.

Acknowledgments. This work is part of the TUM Living Lab Connected Mobility (TUM LLCM) project and has been funded by the Bavarian Ministry of Economic Affairs, Energy and Technology (StMWi) through the Center Digitisation. Bavaria, an initiative of the Bavarian State Government.

References

1. van der Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. MP, pp. 105–128. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14430-6_8
2. van der Aalst, W., Desel, J., Oberweis, A. (eds.): Business Process Management: Models, Techniques, and Empirical Studies. LNCS, vol. 1806. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45594-9
3. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Alonso, G., Saltor, F., Ramos, I. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 467–483. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0101003
4. 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 (2016)
5. Brückmann, T., Gruhn, V., Pfeiffer, M.: Towards real-time monitoring and controlling of enterprise architectures using business software control centers. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 287–294. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23798-0_31
6. Calçado, P.: Building products at soundcloud part III: Microservices in scala and finagle. Technical report, SoundCloud Limited (2014). https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-3-microservices-in-scala-and-finagle
7. Cockcroft, A.: Microservices workshop: why, what, and how to get there (2016)
8. Cuadrado, F., García, B., Dueñas, J.C., Parada, H.A.: A case study on software evolution towards service-oriented architecture. In: 22nd International Conference on Advanced Information Networking and Applications-Workshops, AINAW 2008, pp. 1399–1404. IEEE (2008)
9. Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow. CoRR abs/1606.04036 (2016)
10. Evans, E.J.: Domain-Driven Design: Tacking Complexity in the Heart of Software. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)
11. Fowler, M., Lewis, J.: Microservices. Technical report, ThoughtWorks (2014). https://martinfowler.com/articles/microservices.html
12. Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: trends, focus, and potential for industrial adoption. In: International Conference on Software Architecture (ICSA) (2017)
13. Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., Di Salle, A.: Microart: a software architecture recovery tool for maintaining microservice-based systems. In: IEEE International Conference on Software Architecture (ICSA) (2017)
14. Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., Di Salle, A.: Towards recovering the software architecture of microservice-based systems. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 46–53. IEEE (2017)
15. The Open Group Library: ArchiMate 3.0 Specification. Van Haren Publishing, Zaltbommel (2016)

16. Haitzer, T., Zdun, U.: DSL-based support for semi-automated architectural component model abstraction throughout the software lifecycle. In: Proceedings of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures, pp. 61–70. ACM (2012)
17. Haren, V.: TOGAF Version 9.1, 10th edn. Van Haren Publishing, Zaltbommel (2011)
18. van Hoorn, A., Rohr, M., Hasselbring, W., Waller, J., Ehlers, J., Frey, S., Kieselhorst, D.: Continuous monitoring of software services: design and application of the Kieker framework (2009)
19. van Hoorn, A., Waller, J., Hasselbring, W.: Kieker: a framework for application performance monitoring and dynamic software analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE 2012, pp. 247–248. ACM, New York (2012)
20. Ihde, S.: From a monolith to microservices + REST: the evolution of Linkedin's service architecture (2015). http://www.infoq.com/presentations/linkedin-microservices-urn. Accessed 18 Nov 2017
21. Josephsen, D.: Building a Monitoring Infrastructure with Nagios. Prentice Hall PTR, Upper Saddle River (2007)
22. Kramer, S.: The biggest thing Amazon got right: the platform (2011). https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/. Accessed 18 Nov 2017
23. Montesi, F., Weber, J.: Circuit breakers, discovery, and API gateways in microservices. CoRR abs/1609.05830 (2016)
24. Netflix: Eureka. https://github.com/Netflix/eureka. Accessed 18 Oct 2017
25. Richards, M.: Microservices vs. Service-Oriented Architecture, 1st edn. O'Reilly Media, Inc. (2016)
26. O'Brien, L., Stoermer, C.: Architecture reconstruction case study. Technical report CMU/SEI-2003-TN-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh (2003). http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6431
27. O'Brien, L., Stoermer, C., Verhoef, C.: Software architecture reconstruction: practice needs and current approaches. Technical report CMU/SEI-2002-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh (2002)
28. Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H.A., Mankovskii, S.: Solving big data challenges for enterprise application performance management. CoRR abs/1208.4167 (2012)
29. Sigelman, B.H., Barroso, L.A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., Shanbhag, C.: Dapper, a large-scale distributed systems tracing infrastructure. Technical report Google, Inc. (2010)
30. Toffetti, G., Brunner, S., Blöchlinger, M., Dudouet, F., Edmonds, A.: An architecture for self-managing microservices. In: Proceedings of the 1st International Workshop on Automated Incident Management in Cloud, AIMC 2015, pp. 19–24. ACM, New York (2015)
31. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. **26**(3), 276–292 (1987)

Towards Reliable Predictive Process Monitoring

Christopher Klinkmüller[1(✉)], Nick R. T. P. van Beest[2], and Ingo Weber[1]

[1] Data61, CSIRO, Sydney, Australia
{christopher.klinkmuller,ingo.weber}@data61.csiro.au
[2] Data61, CSIRO, Brisbane, Australia
nick.vanbeest@data61.csiro.au

Abstract. Predictive process monitoring is concerned with anticipating the future behavior of running process instances. Prior work primarily focused on the performance of monitoring approaches and spent little effort on understanding other aspects such as reliability. This limits the potential to reuse the approaches across scenarios. From this starting point, we discuss how synthetic data can facilitate a better understanding of approaches and then use synthetic data in two experiments. We focus on prediction as classification of process instances during execution, solely considering the discrete event behavior. First, we compare different feature representations and reveal that sub-trace occurrence can cover a broader variety of relationships in the data than other representations. Second, we present evidence that the popular strategy of cutting traces to certain prefix lengths to learn prediction models for ongoing instances is prone to yield unreliable models and that the underlying problem can be avoided by using approaches that learn from complete traces. Our experiments provide a basis for future research and highlight that an evaluation solely targeting performance incurs the risk of incorrectly assessing benefits and limitations.

Keywords: Behavioral classification · Predictive process monitoring
Process mining · Machine learning

1 Introduction

One goal of *process mining* [1] is to support organizations in identifying deviations from the expected behavior during process execution. Here, *conformance checking* techniques, e.g., [2,3], analyze, if the current state of a process instance in terms of its *event trace* conforms to the normative behavior. Complementary, *predictive process monitoring* is concerned with anticipating the future development of an instance [4] and focuses on (i) estimating the remaining execution time [5,6]; (ii) determining the next events [6–8]; (iii) measuring the risk associated with possible paths to completion [9,10]; and (iv) predicting the outcome of an instance [11–14].

In this work, we primarily focus on the latter category where *prediction models* are derived from historical data which contains event traces of past instances,

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 163–181, 2018.
https://doi.org/10.1007/978-3-319-92901-9_15

labeled with the classes. Those classes e.g., indicate whether a business constraint was met [11,15], or describe the quality level achieved by the instance [12]. In prior work, a general approach to implementing the prediction is to (i) transform each trace from the historical data into a feature vector; (ii) use standard machine learning algorithms to train a model on the feature vectors and the associated outcome; (iii) represent ongoing traces as features vectors; and (iv) apply the model to obtain a prediction at runtime.

In this context, our work is motivated by the observation that prior work is largely limited to reporting performance metrics obtained on some real-world datasets (see Sect. 2) and to interpreting the results as evidence for the quality of the proposed approach. In the machine learning community, this methodology has long been criticized [17–19], because without further investigation it is unclear whether the performance is due to the discovery of higher level concepts or chance. A recent example in this regard is the study by Jo and Bengio [20]. While convolutional neural networks achieved a high performance for image recognition on various datasets, the study presented evidence that this high performance is partly due to superficial hints in the images which were present in the (training and test) data, but not because the networks derive higher level concepts to describe and recognize objects. Based on this criticism and related empirical evidence, we argue that more emphasis must be put on understanding the circumstances under which certain strategies perform well or poorly, following e.g., [16].

While data from real-world scenarios is certainly important to evaluate and compare the techniques' performance under realistic circumstances, it is often hard to understand to which degree the data enables the learning of a reliable prediction model and to assess whether the techniques learn the correct relationships from the data. In contrast, the generation of synthetic data allows researchers to exert full control over the variables that influence the outcome of a process instance. We therefore discuss and demonstrate how the use of synthetic data as a supplement for real-world data can yield additional insights. In this context and without the loss of generalizability, we focus on the sub-problem of *classifying* traces and only consider categorical outcome variables. Furthermore, we limit our work to *discrete event behavior*, i.e., the ordering of events in terms of sequences of event identifiers. While events can be associated with more information, the ordering of the events is the main dimension that separates predictive process monitoring from other prediction problems. In summary, our contributions are:

1. We present and discuss an approach to generate synthetic data that enables the establishment of an objective baseline, which in turn can be used to gain a more comprehensive understanding of how predictive process mining techniques work.
2. We use this approach to examine two aspects of the classification of traces.
   (a) We analyze different feature encodings for the representation of discrete event behavior and show that our *sub-trace encoding* is better suited for classification of completed traces than encodings from prior work. This

experiment also demonstrates that synthetic data can facilitate the understanding of how approaches pick up relationships between trace properties and outcomes.

(b) We additionally compare two approaches for the classification of ongoing traces. The *local prediction* approach, which works by training a set of classifiers for pre-defined prefix lengths, has often been used in prior work [11,13–15]. We demonstrate that it bears the risk of increasing the importance of irrelevant features. In contrast, a *global prediction* model that relies on a single classifier trained on the completed traces yields a higher reliability. This comparison emphasizes that solely relying on performance measures can be misleading.

The remainder of the paper is organized as follows. Section 2 summarizes related work and Sect. 3 introduces basic definitions. In Sect. 4, we subsequently discuss the process of generating synthetic data and the use of the data for analyzing predictive process mining techniques. Next, Sect. 5 is concerned with comparing different feature encodings. After that, we analyze the local prediction approach in Sect. 6 and the global prediction approach in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Related Work

In the field of *process mining*, comparing event logs with normative process models (i.e. conformance checking [2,3]) or comparing event logs with other event logs (e.g. [21,22]) are common themes. Most of these techniques, however, are offline and focus on full behavior (i.e., partial executions are not taken into account) for the purpose of assessing the differences between normative behavior (represented by models or event logs) and observed behavior (event logs), but not for immediate classification. Partial traces have been analyzed for the purpose of conformance checking (e.g. [23,24]), but these approaches are based on trace alignment, where each event mismatch is taken into account individually and is, therefore, less precise and not complete [3]. In earlier work, we discussed real-time binary conformance checking [25].

The area of *predictive business process monitoring* [4,11] is concerned with forecasting how business process instances will unfold until completion. A first set of approaches aims to predict the next events for a prefix. For example, [6–8] evaluate different neural network architectures for this task. Other approaches focus on estimating the remaining time of a business process instance [5,6,26,27] or until the next event [6,28]. The evaluation of risks arising during execution and threatening the successful completion are studied in [9,10,29].

In the context of this paper, the most important category is the prediction of the outcome of a business process instance. The approaches in [11,13–15,30,31] are what we refer to as *local prediction* approaches, which train prediction models at predefined positions, e.g., at prefixes of lengths 3, 4, 5, etc. As such, they rely on training sets where the outcome is given for the completed traces.

When learning a prediction model at a certain position, the completed traces are reduced to the events that occurred up until this position. We present a *global prediction* approach in [12] where one prediction model is trained on the original training set and applied to all possible partial traces.

Complementary to these approaches, we here provide an analysis which reveals that the local prediction approach is prone to yield unreliable classifiers. Additionally, we demonstrate that global prediction approaches can overcome this limitation. To this end, we discuss an approach that in contrast to [12] also achieves good results for unstructured processes. Moreover, while all the mentioned works limit their evaluation to the presentation of accuracy measures or similar effectiveness metrics, we here additionally study the reliability based on synthetic data that is associated with an objective ground truth. In this context, Metzger and Flöcker [15] studied the filtering of prediction results based on confidence levels. Yet, they relied on a local prediction approach and measured reliability based on the outcome of the prediction model. In contrast, we here assess the reliability with regard to an objective ground truth.

3 Preliminaries

Events and Traces. The execution of a process instance results in *events* that provide information on the current state of the instance. For example, during the processing of a sales order, events might indicate that the order is confirmed, or that the goods were shipped.[1] To capture this information, the observed events are recorded as *traces* in which the events are ordered with respect to the time at which they were observed and described in terms of labels. Events can also be annotated with further information, e.g., with timestamps, resource identifiers, or data that is processed – but we focus solely on the information in a trace that concerns the ordering of events. As mentioned in the introduction, our work can be supplemented with additional features for additional dimensions. A set of traces is grouped in an *event log*.

Definition 1 (Event log, Trace). *Let L be an event log over the set of labels $\mathscr{L}$. Let E be a set of event occurrences and $\lambda : E \to \mathscr{L}$ a labeling function. An event trace $\theta \in L$ is defined in terms of an order $i \in [1, n]$ and a set of events $E_\theta \subseteq E$ with $|E_\theta| = n$ such that $\theta = \langle \lambda(e_1), \lambda(e_2), \ldots, \lambda(e_n) \rangle$. Moreover, given two events $e_i, e_j \in E_\theta$ with $i < j$, we write $e_i \triangleright e_j$ or say that event e_j* follows *e_i.*

During execution, the current state of an instance is given by the events that already occurred. While a trace captures the information of a completed instance, we refer to the sequence of the first l events of an instance as a *prefix*.

[1] The notion of an event used throughout this paper assumes that it corresponds to some activity in an underlying process. System-level logs, for instance, may contain events with some variation in the labels. We assume that, if present, any such variation has been removed in a preprocessing step, and that events belonging to the same activity have the same label.

Definition 2 (Prefix). *Given an event trace* $\theta = \langle \lambda(e_1), \lambda(e_2), \ldots, \lambda(e_n) \rangle$*, a prefix of this trace with length* $l \in \mathbb{N}$ *is defined as*

$$\theta_l := \begin{cases} \langle\rangle & if\ l = 0 \\ \langle \lambda(e_1), \ldots, \lambda(e_l) \rangle & if\ 0 < l \leq n \\ \langle \lambda(e_1), \ldots, \lambda(e_n) \rangle & otherwise \end{cases}$$

For a training log L, we also define the set of all prefixes in this log as $L_{pre} = \{\theta_l | \theta \in L \wedge 1 \leq l \leq |\theta|\}$.

Classification and Features. For each completed trace, the outcome might be given in terms of one of m behavioral classes, where the function $cl : L \rightarrow C$ with $C = \{c_1, \ldots, c_m\}$ represents the classification of the traces. The goal is to learn this classification and to derive a *prediction function* $pr : L_{pre} \rightarrow C$ that can be used to classify traces as well as prefixes. It is important to note that the prediction function is trained on a mapping from the traces to the classes, but allows for the classification of traces and prefixes.

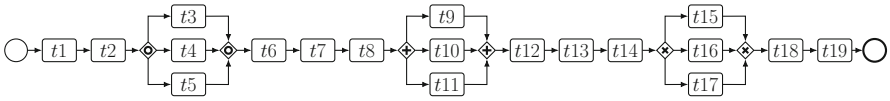
To support the learning of such prediction functions, traces and prefixes are typically encoded using features. More specifically, we are interested in *feature encoding* functions $\phi : L_{pre} \rightarrow \mathbf{F}$ where $\mathbf{F} = (F_1, \ldots, F_n)$ is a vector of n *feature variables*. Thus, prefixes and traces are no longer represented as a sequence of event labels, but as a *feature vector* $\mathbf{f} = (f_1, \ldots, f_n)$ where each element f_i takes a value from the *domain* D_i of the respective feature variable F_i. By relying on a certain feature encoding ϕ, the argument of the prediction function $pr : L_{pre} \rightarrow C$ changes and the function $pr : \phi(L_{pre}) \rightarrow C$ now classifies traces based on feature vectors. Moreover, the learning of such a prediction function is carried out in two steps. The first (optional) step is *feature learning:* given a set of training traces L, the feature encoding function ϕ is learned and each training trace $\theta \in L$ is encoded as a feature vector based on the learned encoding function $\mathbf{f} = \phi(\theta)$. After that, a *prediction model* is learned based on the feature vectors. To classify an observed prefix θ_l^r at runtime, it is first encoded as a feature vector $\mathbf{f}^r = \phi(\theta_l^r)$, which is subsequently passed to the prediction model to obtain the classification $c^r = pr(\mathbf{f}^r)$.

4 Generating Synthetic Data

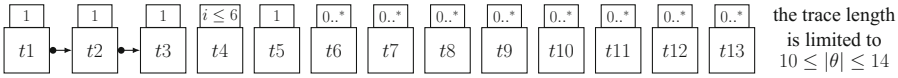
Subsequently, we outline how we generated and used data in our experiments, and point to the aspects that allowed us to gain a deeper understanding of how approaches work. We also made the synthetic data from our experiments publicly available[2].

Generation. To represent a broad variety of scenarios, we generated two datasets. Each dataset is based on one process model. These models are oriented towards the distinction between "Lasagna" and "Spaghetti" processes [1].

[2] https://doi.org/10.4225/08/5acff82350c74.



(a) Low variability model (modeled in BPMN)



(b) High variability model (modeled in DECLARE[2] [32])

Fig. 1. Models with normative behavior (We introduced the notation $i \leq x$ with $x \in \mathbb{N}$ to indicate that an event has to be among the first x events.)

The *low variance* ("Lasagna") model is on one end of the structuredness spectrum. It imposes strict ordering rules, is mostly sequential, and describes a small set of distinct traces. On the contrary, the *high variance* ("Spaghetti") model allows for a more flexible process execution and enforces only a few constraints resulting in a large amount of possible distinct traces. These two models define the respective normative behavior and are presented in Fig. 1.

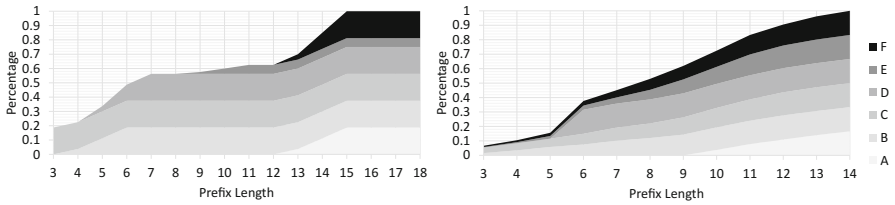
Based on these models, we generate traces and assign them to different behavioral classes by applying a procedure that is inspired by *mutation testing* [33] – a well-known technique for measuring the quality of software test suites. That is, we defined five different mutation operators, which we applied independently to the basic model in order to obtain one model per operator. Table 1 presents the operator descriptions, their class labels (class label A is used for the normative model), and how they were applied to the two models. While these operators represent basic differences in the discrete event behavior, it is conceivable to define more complex operators that also modify other event attributes e.g., the distribution of processing time.

Table 1. Mutation operators and their application to the models

Description	Specific application to models		Class label
	Low variability	High variability	
Change the order of activities	Swap $t6$ and $t8$	Swap $t1$ and $t3$	B
Insert a new activity	Add $t20$ after $t2$	Add $t14$ somewhere	C
Move an activity	Move $t7$ after $t12$	Move $t4$ to $i > 6$	D
Remove an activity	Delete $t9$	Delete $t2$	E
Execute an activity twice	Insert $t14$ after $t14$	Let $t5$ occur twice	F

Subsequently, we simulated the models to obtain a set of distinct traces. For the low variance models, we considered all of the distinct traces that each model defines. This resulted in 90 traces for class E and 270 for each of the other classes. For the high variance model, where the amount of possible traces is very large, we generated 250 distinct traces per class. In the traces, the execution of an activity is represented by a single event whose label is identical to the activity label. Note that, in addition to the structuredness, both datasets also differ with regard to the coverage of the training data relative to the trace space.

Prefix Classifiability. The advantage of the synthetic data is that we do not only know the outcome (i.e., the class) for the entire trace, but we can use the definition of the mutation operators to setup a *classifiability oracle* which constitutes the objective ground truth. This oracle can tell us for each prefix, if the outcome can already be predicted at this point, and if that is the case, what the outcome is. For example, consider the following prefixes of length 3 on the low variability dataset: $\theta_3 = \langle t1, t2, t20 \rangle$ and $\theta'_3 = \langle t1, t2, t3 \rangle$. For θ_3 it can clearly be determined that it belongs to class C, as it contains the event $t20$ which only occurs in traces of this class. Similarly, we can rule out that θ'_3 belongs to C, as it does not contain $t20$ which needed to appear before $t3$. Yet, θ'_3 is still *unclassifiable*, as all remaining classes are still possible for θ'_3. Figure 2 depicts the percentage of classifiable prefixes per class, prefix length, and dataset.



(a) Low variability (b) High variability

Fig. 2. Percentage of classifiable traces per prefix length and class

Observation 1: In our datasets, the percentage of classifiable prefixes overall as well as per class increases monotonically with a growing prefix length.

In addition to this observation, Fig. 2 also shows that on the low variability dataset there are windows of prefix lengths in which traces belonging to a certain class become classifiable, e.g., class D becomes classifiable for prefix lengths between 4 and 7. Yet, the high variability dataset demonstrates that this does not necessarily need to happen. Due to a greater flexibility in the ordering of events, these windows cover the entire range of prefix lengths.

The ground truth of the classification oracle enables us to go beyond an accuracy assessment and to analyze the reliability of correct predictions. In particular,

we can check, if a correctly classified prefix could actually already be classified, or if the classifier was lucky.

Evaluation Metrics. Each set is used once as a test set for the functions that were trained on the union of the remaining trace sets. When evaluating different combinations of feature encodings and prediction functions, we perform a 5-fold cross validation where the event log is split into 5 equally sized sets of traces; training is performed on 4 of the 5 sets, testing against the remaining set. The overall effectiveness reported is the average over all trace sets. As the datasets only contain distinct traces, our evaluation is based on out-of-sample forecasts. That is, a trace is either used for training or testing and it is hence not sufficient for an approach to memorize the data. Instead, it needs to infer higher level relationships and concepts from the data to achieve a high performance.

In the evaluation, we measure the effectiveness of a prediction function for a certain position l (number of events after which we want to predict the outcome) with regard to different confidence levels. In general, the higher the confidence the more certain is the prediction function. At this point, we abstract from the specific confidence measurement and provide more details in the context of the experiments. Given a test trace set T, we obtain the prediction along with the confidence for all prefixes of length l. Next, we determine the set T_{cf} which contains all prefixes where the confidence is equal to or higher than the fixed confidence level. Based on the set of correctly predicted traces $T_{co} \subseteq T_{cf}$, the *precision* $pc = \frac{|T_{co}|}{|T_{cf}|}$ is the ratio of correctly classified and confidently classified prefixes, while the recall $rec = \frac{|T_{co}|}{|T|}$ is the overall share of correctly classified prefixes. In addition to these measures, we also take advantage of the classification oracle and investigate if it supports the correct predictions – i.e., if the predictions are justified given the observed prefix. In particular, we determine the set T_{cl}, which comprises all prefixes in T_{co} that were actually classifiable. This is expressed as the *justification score* $js = \frac{|T_{cl}|}{|T_{co}|}$, i.e., the share of the correct predictions that are justified according to the classification oracle.

5 Feature Encodings

In this section, we compare and analyze five feature encodings, which include four encodings from prior work:

Event occurrence (EvOcc) [13]**:** Each distinct event label is represented as a feature variable. In the feature vector for a trace, an element is set to 1 if the respective label occurs in the trace, and to 0 otherwise.

Event frequency (EvFreq) [13]**:** Again, there is one feature variable per distinct event label. For a given trace, the vector elements contain the number of occurrences of the respective label in the trace.

Simple index (Index) [13]**:** Given the maximal trace length l_m, there is one feature variable for each position $1 \leq p \leq l_m$. For a trace, each vector element

contains the event label from the respective position. If a trace is shorter than the maximal trace length $|\theta| < l_m$, the elements for positions $p > |\theta|$ are set to a default label to indicate the absence of a label.

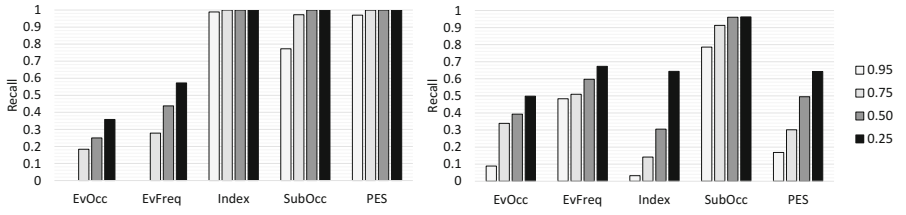
Prime event structures (PES) [12]**:** For each behavioral class, a real-valued feature variable is introduced. Here, a variable represents a matching score that indicates the degree to which a trace resembles the behavior of the respective class. To this end, for each class the according traces from the training log are transformed into a PES that captures the behavioral relations between the event labels, along with the execution frequencies for multiple exclusive branches. Subsequently, a weighted matching score is obtained for each PES, based on the identified behavioral differences with the trace and the branching frequencies of the matched trace in that PES.

Sub-trace occurrence (SubOcc): Here, we consider binary feature variables that indicate whether a sub-trace occurred in a trace. A sub-trace ς is a sequence of labels $\varsigma = \langle \lambda_1, \ldots, \lambda_o \rangle$ and we say that a sub-trace occurs in a trace $\varsigma \in \theta$, if all labels in ς occur in the same order as in θ, i.e., if $\forall 1 \leq i < j \leq |\varsigma| : \exists e_k, e_l \in E_\theta : \varsigma_i = \lambda(e_k) \wedge \varsigma_j = \lambda(e_l) \wedge e_k \triangleright e_l$. To establish the feature variables, we derive the set of sub-traces from the training traces by applying sequence mining and in particular the SPADE algorithm [34]. We control the identification of relevant sub-traces by two parameters. First, we only consider sub-traces with a support of at least .05, i.e., the sub-traces have to occur in at least 5% of the training prefixes. Second, we restrict the length of the sub-traces to 3 which we found to be a sufficient length for our datasets. For each sub-trace in the mined set, there is one feature variable. For a trace, a vector element is set to 1 if the corresponding sub-trace occurs in the trace, and to 0 otherwise. This encoding shares similarities with declarative process discovery algorithms like DECLARE [32] that aim to extract general execution constraints. Such constraints specify, for instance, that two activities are sequential, or activities are executed in a loop, etc. However, we here abstain from this generalization and focus on the concrete behavior. As such we can, for example, capture how often a loop is executed, and similar differentiations which may be important to distinguish classes.

We evaluated these encodings by adopting the experimental setup from [13, 14] and chose Random Forest[3] [35] to learn the prediction functions. At heart, a Random Forest trains a set of decision trees of size *ntree*. Each of these trees is trained on a random sample drawn from the training dataset with replacement. Moreover, not all features, but only a randomly drawn sub-set of features of size *mtry* is considered when a decision node is induced. To discriminate an object, each tree classifies it and the overall result is the class that was yielded by the most trees. We interpret the percentage of the trees that voted for this class as the confidence in the result. Additionally, we followed guidelines from the literature [35,36] and focused on optimizing *mtry*, while we used a default value of

[3] We use the default implementation provided by the `randomForest` package for `R` (https://cran.r-project.org/package=randomForest, accessed: 15/11/2017).

500 for *ntree*. That is, in each iteration of the 5-fold cross-validation that we use to evaluate the effectiveness, we performed another 5-fold cross validation with 3 repetitions on the training set to evaluate different values for *mtry* and chose the one with the highest accuracy to obtain the final Random Forest[4]. Finally, we determined the recall for all encodings and for four different confidence levels (95%, 75%, 50%, 25%). We do not consider the precision and justification score here, as we are evaluating the feature encodings over the completed traces and can thus be sure that there is evidence that supports the classification of a trace. Figure 3 summarizes the recall for each encoding.



(a) Low variability (b) High variability

Fig. 3. Accuracy of the different encodings per confidence level

Observation 2: The sub-trace occurrence yields the highest accuracy.

On both datasets, the EvOcc and EvFreq encodings yield low accuracy values, as they discard information regarding the ordering of events and hence struggle to provide meaningful features in cases where the event ordering is relevant for classification. The Index encoding achieves a perfect accuracy on the low variability dataset, but performs poorly on the high variability dataset because it does not enable a classifier to infer that events need to be positioned relative to each other. For example, consider that event $t2$ has to eventually follow event $t1$, in order for a trace to belong to a certain class. Here, the trees in the Random Forest need to introduce one rule per possible combination of absolute positions that satisfy this characteristic, e.g., $\lambda_1 = t1 \wedge \lambda_2 = t2$, $\lambda_1 = t1 \wedge \lambda_3 = t2 \ldots$, $\lambda_2 = t1 \wedge \lambda_3 = t2, \ldots$. Thus, the higher the variance in the events' positions and the trace length, the more data is needed to infer all these rules. The PES encoding suffers from the same problem, because the approach as presented in [12] focuses on differences in specific positions in the trace and is, as such, more suited for structured processes as it fails to provide a more flexible representation of the behavior in the presence of variability. Finally, the SubOcc provides the richest features, because it also includes relative relationships between events and thus allows the classifier to consider these relationships independent of the

[4] In particular, we applied the random search strategy from the `caret` package for `R` (https://cran.r-project.org/package=caret, accessed: 15/11/2017).

absolute positioning. For low confidence intervals, it achieves a recall of 1 on the low and of .96 on the high variability dataset. Yet, the recall drops with an increase in the confidence level on both datasets. This is due to the high number of features and the random sampling of traces as well as features in the Random Forest, which can result in some irrelevant features distorting the classification.

Conclusion 1: Encoding relative ordering relationships between events as features can improve the effectiveness of prediction models.

6 Local Prediction

Next, we examine the local prediction approach, where one prediction function pr_l is trained for each prefix length l that was (manually) determined beforehand. To induce such a function from the training data, all traces in the dataset are reduced to prefixes of length l. At runtime, the prediction function that matches the number of observed events is used for classification.

Table 2. Illustrative example training log

Class	Traces			
A	$\langle t1, t2, t3, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$	$\langle t1, t3, t2, t4, t5\rangle$
B	$\langle t1, t2, t3, t4, t5, t6\rangle$	$\langle t1, t3, t2, t4, t5, t6\rangle$	$\langle t1, t2, t3, t4, t6, t5\rangle$	$\langle t1, t6, t2, t3, t4, t5\rangle$

For illustration, Table 2 depicts an example training log that contains eight traces which are assigned to one of two classes. In class A we observe that $t1$ always occurs first, followed by $t2$ and $t3$ in any order, and ending with event $t4$ followed by $t5$. Class B shows the same behavior, but there is an additional event $t6$ which seems to occur at an arbitrary position. Clearly, the existence of $t6$ is a perfectly discriminating characteristic. Now, consider that we learn a prediction model at position 4 and only consider the prefixes of length 4. Then, $t6$ only occurs in one out of four prefixes in class B, but the sub-trace $\langle t2, t3\rangle$ occurs three times in class B and only once in class A. Thus, a classifier would deem this sub-trace more important for the classification than the existence of $t6$. This shows that by removing the suffixes, we change the importance of features and in the worst case favor irrelevant over relevant features.

To further examine this risk, we evaluate the local prediction approach. That means, for each prefix length greater than 2, we repeat the procedure from Sect. 5 and evaluate the Random Forest in combination with the SubOcc encoding. In addition to the *classifiability oracle* (Oracle Cl.), we also use the *classifiability oracle with a default option* (Oracle Def.) as a baseline: for unclassifiable prefixes this oracle predicts normative class A in the absence of evidence to the contrary. That may be desirable, for instance, in settings where the normative case is much more frequent than any deviation. We then compare the Oracles' recall values

to the recall values of the classifiers, to see how close the classifiers come to the objective ground truth. Figure 4 shows the effectiveness yielded at the different prefix lengths for four confidence levels.

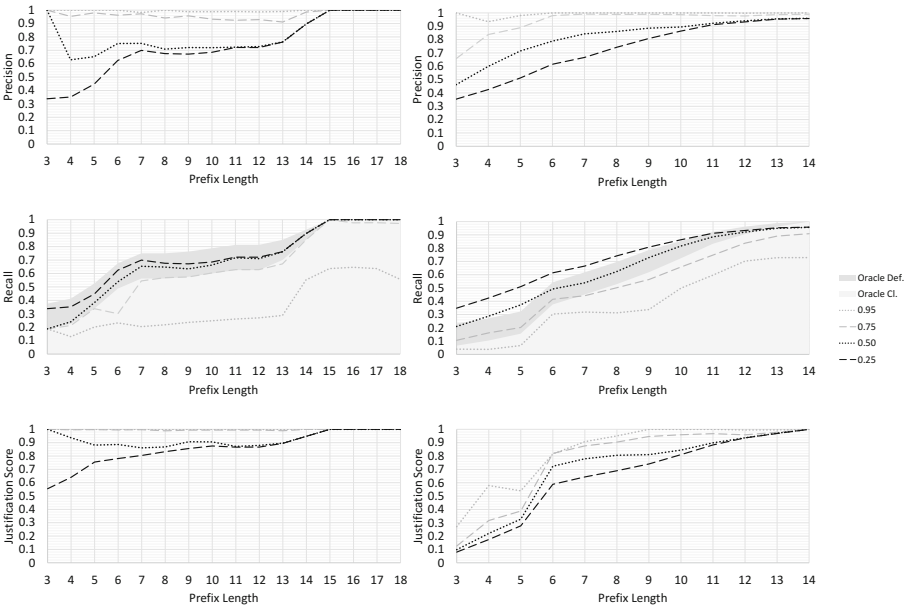


Fig. 4. Prefix length vs. precision, recall and justifiability score on the low (left) and high variability datasets (right)

Observation 3: Raising the confidence (i) increases the precision and justification score; (ii) decreases the recall; but (iii) does not ensure that all correct predictions are justified.

The positive relation between the confidence and the precision as well as the justification score and the negative relation between the confidence and the recall are expected, as a higher confidence level requires more agreement on the result. Interestingly, increasing the confidence does not guarantee a high justification score. In fact, independent of the confidence level the justification score is quite low at the beginning and only approaches 1 at high prefix lengths on the high variability dataset. A similar effect can be observed on the low variability dataset. This implies that for a large portion of correct predictions the classifiers rely on irrelevant features and exploit spurious correlations.

Observation 4: The classifiers' recall can exceed that of the oracles.

On the high variability dataset the classifiers outperform the oracles for low confidence levels and prefix lengths. A performance better than Oracle Cl. can be explained by relying on a default option. However, the improved accuracy in comparison to Oracle Def. substantiates that the classifiers are exploiting spurious correlations.

Observation 5: The recall might decrease with a growing prefix length, even if the justification score is not decreasing.

For example, on the low variability dataset the accuracy drops after position 7 for confidence levels of 25% and 50% or at position 18 for a confidence of 95%. In both cases, the justification score remains stable or increases. A decreasing effectiveness for higher prefix lengths can also be observed in prior work, e.g., in the evaluation in [13]. This indicates that the classification rules at a certain prefix length are in part invalidated at a higher prefix length and that the importance of some irrelevant features decreases with a growing prefix length.

Conclusion 2: Our experiments provide clear evidence that the local prediction approach can provide predictions that are not justified by the data observed, which may be due to an overly high reliance of irrelevant features. This negatively impacts the reliability of the prediction models and thus the practical applicability.

7 Global Prediction

In this section, we test our hypothesis that a global prediction approach can overcome the problems of the local prediction approach. To this end, we first introduce our specific implementation of the global prediction approach and afterwards present the evaluation results.

In contrast to the local prediction approach, we only learn one classifier in the global prediction approach, which is then used to classify any prefix. Moreover, this classifier is trained solely on the completed traces and their associated outcomes. We utilize rule-based classification and in particular a variant of the RIPPER algorithm [37], since a direct application of classification approaches like the Random Forest does not yield high-quality results in this setting (as explained below). That is, the classifier consists of a set of rules of the form `IF condition THEN` c where `condition` defines properties that a prefix needs to possess in order to belong to class c. Given a training log, a set of rules is established based on the following two step procedure.

First, we define the set of elementary conditions that are considered during rule induction. To this end, we start with mining all sub-traces as explained in Sect. 5. For each sub-trace ς, there is one *sub-trace condition* that evaluates to true, if the sub-trace is part of the prefix. These conditions enable us to check whether an event occurs at all or relative to other events (i.e., before or after them). However, sometimes it is the absence of events that is representative for

a certain behavioral class. Therefore, for each ς we introduce *absence conditions*. Such a condition evaluates to true if the prefix reached a length and the sub-trace did not occur beforehand. To this end, we iterate over all event labels and for each event label λ we determine its preset, i.e., the set of labels that solely occur before λ. Next, we group all events that have the same preset. In the preset comparison, we remove those labels from the presets that we are currently comparing. We then yield the set of absence conditions by combining each of the obtained groups with each sub-trace. In the evaluation of an absence condition, we check if any of the position labels is part of the prefix. If that is the case, we check if the sub-trace does not occur before the first label in the prefix. Lastly, we cluster perfectly correlated conditions into an elementary or-condition that evaluates to true if any of the sub-conditions are satisfied. Two conditions are perfectly correlated, if for all training traces the conditions evaluate to the same value. The reason to cluster such conditions is that we want to ensure that we decide on the class as early as possible. For example, consider the case where an event $t1$ needs to occur before two events $t2$ and $t3$ which always occur together but in any order. For the classification of completed traces it would not matter, if we consider $\langle t1, t2\rangle$ or $\langle t1, t3\rangle$ for representing this correlation. Yet, when we classify running instances and only consider $\langle t1, t2\rangle$, we will lately classify prefixes where $t3$ occurs before $t2$. This is where the Random Forest fails: it just picks one of the conditions at random, which can severely delay the classification.

Second, in the rule induction we process each behavioral class independently. For a given class c, we first extract the set of traces T_c from the training log that belong to c and the set of traces $T_{\not c}$ that do not belong to c. We then iteratively learn one rule and after that remove all traces from T_c for which the rule is satisfied. We repeat this step until all traces in T_c are covered by at least one rule.

In each iteration, we learn a rule of the form $\texttt{condition}_1$ `AND` ... `AND` $\texttt{condition}_n$ based on a beam search. That is, we first select the elementary condition that best separates the traces in T_c from those in $T_{\not c}$. Next, we select another elementary condition and combine it with the first elementary condition, if it improves the discriminative power of the rule. We keep adding elementary conditions until the discriminative power of the rule is not improved any further. To measure the discriminative power of a condition we employ the *FOIL gain* [38], which considers not only the overall amount of traces in T_c and $T_{\not c}$ for which the condition is satisfied, but also the ratio of such rules in both T_c and $T_{\not c}$.

At runtime, we classify a prefix or trace by selecting all rules that are satisfied for the prefix. If no rule is satisfied, we do not predict an outcome. Otherwise, we select the most confident rule and return the respective class. The confidence is the percentage of training traces that belong to the rules' class among the set of training traces for which the rule condition is satisfied. Note that in the evaluation we do not obtain a class if the confidence of the selected rule is too low. Figure 5 shows the evaluation results.

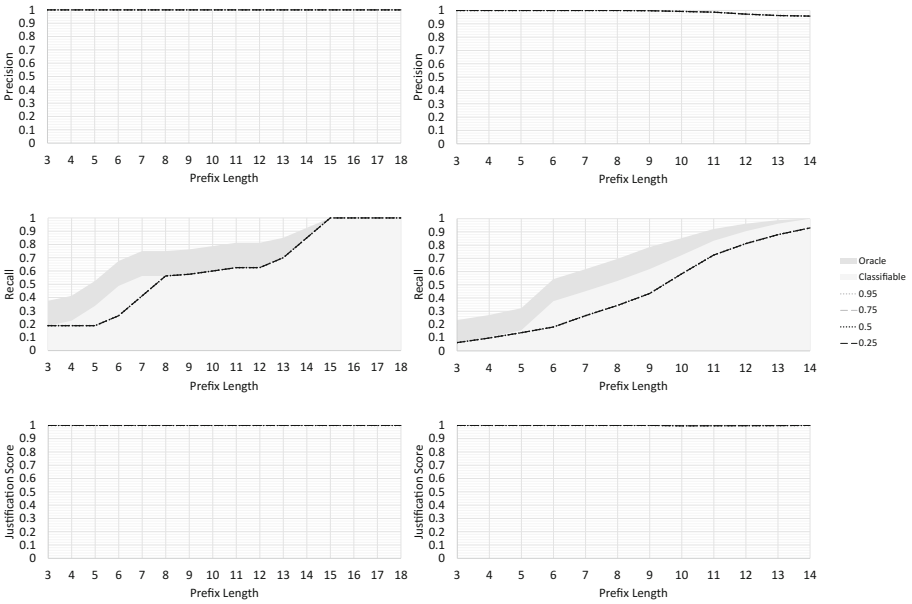


Fig. 5. Prefix length vs. precision, recall and justifiability score on the low (left) and high variability datasets (right). Note that the same results were yielded for all confidence levels.

Observation 6: The confidence level does not impact the effectiveness and reliability of the global prediction approach.

Contrary to the local prediction approach, the confidence level has no impact on the effectiveness. This indicates that global prediction models are reliable indicators which exploit relationships between the relevant features and the outcome.

Observation 7: The precision and the justification score are virtually perfect for all prefix lengths.

In stark contrast to the local prediction approach, the precision and the justification score are 1 for all prefixes, with one exception on the high variability dataset where the precision is slightly lower for prefix lengths greater than 10. This confirms that the results of the global approach are justified.

Observation 8: The recall development is similar to, but at points lower than, that of the classifiability oracle.

The lower recall is due to the simplistic nature of the rule induction algorithm, the relative nature of the sub-trace encoding, which does not consider the absolute positioning of events and (on the high variability dataset)

the high variance in the events positions' due to which there is an extensive amount of possible conditions. Yet, over the completed traces the global approach yields a performance that is virtually equal to the local prediction approach. For smaller prefix lengths, the recall falls behind that of the local prediction approach, which is due to the high percentage of unjustified predictions that the local prediction approach yields. In summary, we thus conclude:

Conclusion 3: Global prediction approaches can avoid the risk of increasing the importance of irrelevant features and hence yield reliable results.

8 Discussion

In this paper, we examined the prediction of the outcome of process instances during execution, especially classification of discrete event behavior. With regard to the design of such prediction approaches, the findings of our systematic evaluation are threefold. First, encodings for representing running process instances should consider the relative positioning of events, as shown by the effectiveness of the sub-trace encoding over completed traces. Second, training prediction models at fixed positions entails the risk of establishing spurious correlations between irrelevant features and the outcome. Third, in order to avoid that risk, the prediction models should be learned based on the *completed traces rather than* reducing the traces to *prefixes*.

In addition, we demonstrated that the evaluation of predictive process monitoring approaches can benefit from synthetic data. Admittedly, the most critical threat to the validity of the experiments based on synthetic datasets pertains the external and ecological validity. However, we argue that synthetic data enables a thorough analysis of the underlying principles, which goes beyond the common effectiveness-driven evaluation in prior work. To repeat such an analysis with a real-world case study would require an enormous effort to control all influential factors and fully understand the reasons for differences. This is demonstrated especially by the negative results regarding local classification. Here, our results can be seen as *counter-examples*, which without doubt pinpoint a real limitation of the local prediction approach.

For future research, we propose two main directions. First, although our results support the usage of the sub-trace encoding along with a global prediction approach, they also suggest that the presented approach can be improved further. Second, research and practice would benefit from a more elaborate set of tools and methods that support the performance and reliability assessment, especially on real-world data.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19345-3
2. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
3. García-Bañuelos, L., van Beest, N.R.T.P., Dumas, M., La Rosa, M., Mertens, W.: Complete and interpretable conformance checking of business processes. IEEE Trans. Softw. Eng. **44**(3), 262–290 (2018)
4. Dumas, M., Maggi, F.M.: Enabling process innovation via deviance mining and predictive monitoring. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. MP, pp. 145–154. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14430-6_10
5. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_27
6. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
7. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**(August), 129–140 (2017)
8. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 252–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_15
9. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_8
10. Conforti, R., Fink, S., Manderscheid, J., Röglinger, M.: PRISM – a predictive risk monitoring approach for business processes. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 383–400. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_22
11. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
12. van Beest, N.R.T.P., Weber, I.: Behavioral classification of business process executions at runtime. In: Dumas, M., Fantinato, M. (eds.) BPM 2016. LNBIP, vol. 281, pp. 339–353. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58457-7_25
13. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21

14. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 401–417. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_23
15. Metzger, A., Föcker, F.: Predictive business process monitoring considering reliability estimates. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 445–460. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_28
16. Aha, D.W.: Generalizing from case studies: a case study. In: ICML (1992)
17. Cohen, P.R., Jensen, D.: Overfitting explained. In: AISTATS (1997)
18. Salzberg, S.L.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov. **1**(3), 317–328 (1997)
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. **7**, 1–30 (2006)
20. Jo, J., Bengio, Y.: Measuring the tendency of CNNs to learn surface statistical regularities. CoRR abs/1711.11561 (2017)
21. van Beest, N.R.T.P., Dumas, M., García-Bañuelos, L., La Rosa, M.: Log delta analysis: interpretable differencing of business process event logs. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 386–405. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_26
22. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 406–422. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_27
23. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: Online conformance checking: relating event streams to process models using prefix-alignments. Int. J. Data Sci. Anal. 1–16 (2017). https://doi.org/10.1007/s41060-017-0078-6
24. Burattin, A.: Online conformance checking for petri nets and event streams. In: BPM 2017 Demo Track (2017)
25. Weber, I., Rogge-Solti, A., Li, C., Mendling, J.: CCaaS: online conformance checking as a service. In: BPM, Demo Track (2015)
26. van der Aalst, W., Schonenberg, M., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
27. Metzger, A., Leitner, P., Ivanovic, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., Pohl, K.: Comparing and combining predictive business process monitoring techniques. IEEE Trans. SCM Syst. **45**(2), 276–290 (2015)
28. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: Intra and inter-case features in predictive process monitoring: a tale of two dimensions. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 306–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_18
29. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 211–216. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_22
30. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 361–376. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_22

31. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Trans. Serv. Comput. (2016, in press)
32. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM (2011)
33. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE TSE **37**(5), 649–678 (2011)
34. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1), 31–60 (2001)
35. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
36. Hastie, T., Tibshirani, R., Friedman, J.: Random forests. The Elements of Statistical Learning. SSS, pp. 587–604. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7_15
37. Cohen, W.W.: Fast effective rule induction. In: ICML (1995)
38. Quinlan, J.: Learning logical definitions from relations. Mach. Learn. **5**(3), 239–266 (1990)

Extracting Object-Centric Event Logs to Support Process Mining on Databases

Guangming Li[1(✉)], Eduardo González López de Murillas[1], Renata Medeiros de Carvalho[1], and Wil M. P. van der Aalst[1,2]

[1] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{g.li.3,e.gonzalez,r.carvalho,w.m.p.v.d.aalst}@tue.nl
[2] RWTH Aachen University, Aachen, Germany

Abstract. Process mining helps organizations to investigate how their operational processes are executed and how these can be improved. Process mining requires event logs extracted from information systems supporting these processes. The eXtensible Event Stream (XES) format is the current standard which requires a case notion to correlate events. However, it has problems to deal with object-centric data (e.g., database tables) due to the existence of one-to-many and many-to-many relations. In this paper, we propose an approach to extract, transform and store object-centric data, resulting in eXtensible Object-Centric (XOC) event logs. The XOC format does not require a case notion to avoid flattening multi-dimensional data. Besides, based on so-called object models which represent the states of a database, a XOC log can reveal the evolution of the database along with corresponding events. Dealing with object-centric data enables new process mining techniques that are able to capture the real processes much better.

1 Introduction

Process mining represents a set of techniques which are widely used to extract insights from event data generated by information systems. The starting point for process mining techniques is formed by event logs. The XES log format [1] is widely employed to generate event logs. In general, a XES log consists of a collection of traces. A trace describes the life-cycle of a particular case (i.e., a process instance) in terms of the activities executed. Process-aware information systems (e.g., BPM/WFM systems) which assume an explicit case notion to correlate events, provide such logs directly.

However, the information systems one encounters in most organizations are *object-centric*. Some examples of these systems are Customer Relationship Management (CRM) and/or Enterprise Resource Planning (ERP) which provide business functions such as procurement, production, sales, delivery, finance, etc. Examples are the ERP/CRM suites from vendors such as SAP (S/4HANA), Oracle (E-Business Suite), Microsoft (Dynamics 365), and Salesforce (CRM). There

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 182–199, 2018.
https://doi.org/10.1007/978-3-319-92901-9_16

are also some free and open source alternatives such as Dolibarr and Odoo. A common feature of these systems is that they are built on top of database technology, i.e., they contain hundreds of tables covering customers, orders, deliveries, etc. Figure 1 shows a fragment of data generated by the Dolibarr ERP system. This is an example of object-centric data since object instances are explicitly recorded (e.g., orders, which can be abstracted as objects) while events related to the underlying process are implicitly recorded by other means, e.g., through redo logs (cf. Table 1). Approaches to extract XES logs from databases with the aid of ontology [3] or redo logs [5] have been previously proposed.

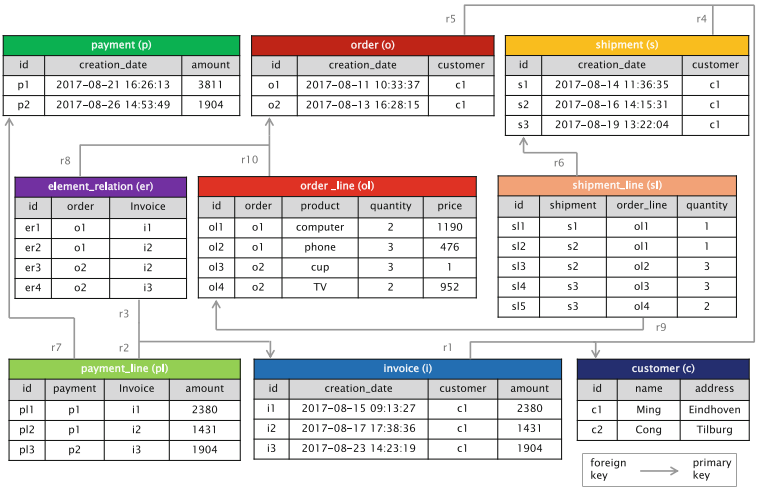


Fig. 1. A fragment of data generated by a real ERP system.

Although there exist approaches to extract XES logs from object-centric data, the constraints set by this format do not match the nature of the data at hand. The following challenges have been identified when applying the XES format to object-centric data:

- *It is required to identify the case id for the whole process.* Database object-centric data lacks a single explicit definition of case notion. On the contrary, because of the existence of multiple classes of objects (tables) and relations (foreign keys), object-centric data can be correlated in many ways. Each of these correlations may correspond to different processes affecting the same data, or different views on the same process (customer vs. provider point of view). In the case of XES, a case notion has to be defined beforehand to proceed with the extraction, requiring one dedicated event log for each perspective to be analyzed.
- *The quality of the input data gets compromised.* If we straightjacket object-centric data into XES logs, the input data with one-to-many and many-to-many relations is flattened into separate traces, in which events referred to by

multiple cases are duplicated. This forced transformation introduces unnecessary redundancy and leads to problems such as data convergence and divergence [9].[1]

– *Interactions between process instances get lost.* Traces in XES logs only describe the evolution (i.e., lifecycle) of one type of process instance and they are typically considered in isolation. Due to the lack of interactions between process instances, XES logs cannot provide a whole view to indicate the state of a system.
– *The data perspective is only secondary.*[2] The XES format focuses on the behavioral perspective, considering any additional information as trace or event attributes. In this sense, the information not directly related to the events is discarded (records, data objects, etc.), which weakens the data perspective of the original system.

To face the problems mentioned above, we propose a novel log format to organize object-centric data from databases, resulting in *eXtensible Object-Centric (XOC*[3]*)* logs. This log format provides an evolutionary view on databases based on the idea that a log is a list of events and each event refers to an *object model* (cf. Sect. 2.1) representing the state of the database just updated by the event. Process mining takes logs as input, and this log format transforms the raw data from databases into event logs, which paves a road to analyze databases in a process mining approach.

Compared with XES, the XOC format has the following contributions. It correlates events based on objects rather than a case notion, which solves the challenge of identifying a case id. Besides, without the constraints set by the case notion, XOC logs can better deal with one-to-many and many-to-many relations. Additionally, more powerful concepts such as *object models* are employed to provide a whole view of the state of the database as well as the interactions between instances. Moreover, all columns in a record can be abstracted as attributes of an object (denoting the record) to enrich the data perspective. To validate the log format, we present an approach to automatically transform object-centric data into XOC logs. In resulting logs, an event type is defined on the information system level (e.g., "create order") rather than the database level (e.g., "insert an order record" and "insert an order line record") [5], which makes extracted events more intuitive for users. Besides these contributions, XOC logs enable new process mining techniques to further solve the problems mentioned above. Discovering data-aware process models like the *object-centric behavioral constraint (OCBC)* models [8], to better reveal the complex interactions between the behavioral and data perspectives (see footnote 2) (cf. Fig. 5), checking con-

[1] Data Convergence means the same event is related to multiple process instances at once. Data divergence means that multiple events of the same activity are related to one process instance, i.e., the same activity is performed multiple times for the same process instance (cf. Fig. 4).

[2] The behavioral perspective refers to the control-flow of events while the data perspective refers to data attributes.

[3] XOC is pronounced as /sɒk/.

formance for deviations which cannot be detected by conventional methods [14] and predicting future events and objects according to a discovered OCBC model (cf. Sect. 4) are some examples of enabled techniques.

The remainder is organized as follows. In Sect. 2, we describe the XOC log format. Section 3 proposes an approach to extract XOC logs from databases, and Sect. 4 demonstrates an implementation of the approach and a comparison between XOC logs and XES logs. Section 5 reviews the related work while Sect. 6 concludes the paper.

2 Object-Centric Event Log

Process mining techniques take event logs as input. In order to apply these techniques to object-centric data, we propose a novel log format to organize such data in this section.

2.1 Object-Centric Data

Increasingly, organizations are employing object-centric information systems, such as ERP and CRM, to deal with their transactions. In this paper, we use the term "object" to abstract data elements (e.g., records in database tables) generated by information systems. In this sense, object-centric systems refer to systems which record the transactions of the same category (e.g., orders) in the same table (e.g., the "order" table) based on the relational database technology. Accordingly, the data generated by such systems are called object-centric data. Figure 1 shows an example of object-centric data.

Objects are grouped in classes and have some attributes. For example, a record in the "order" table (e.g., the first row) can be considered as an object of class "order". Each value (e.g., "c1") in the record can be considered as an attribute of the object. Besides, there exist class relationships between classes, which corresponds to dependency relations between tables. For instance, there is a class relationship between class "order" and class "order line", indicated by one of foreign keys of table "order_line", which refers to the primary key of table "order".

Definition 1 (Object Model). *Let $\mathcal{U}_O$ be the universe of* objects*, $\mathcal{U}_C$ be the universe of* classes*, $\mathcal{U}_{RT}$ be the universe of* class relationships*, $\mathcal{U}_{Attr}$ be the universe of* attribute names *and $\mathcal{U}_{Val}$ be the universe of* attribute values*. $C \subseteq \mathcal{U}_C$ is a set of classes and $RT \subseteq \mathcal{U}_{RT}$ is a set of class relationships. An object model is a tuple $OM = (Obj, Rel, class, objectAttr)$, where*

- *$Obj \subseteq \mathcal{U}_O$ is a set of objects,*
- *$Rel \subseteq RT \times Obj \times Obj$ is a set of object relations,*
- *$class \in Obj \rightarrow C$ maps objects onto classes, and*
- *$objectAttr \in Obj \rightarrow (\mathcal{U}_{Attr} \not\rightarrow \mathcal{U}_{Val})$ maps objects onto a partial function assigning values to some attributes.*

$\mathcal{U}_{OM}$ is the universe of object models.

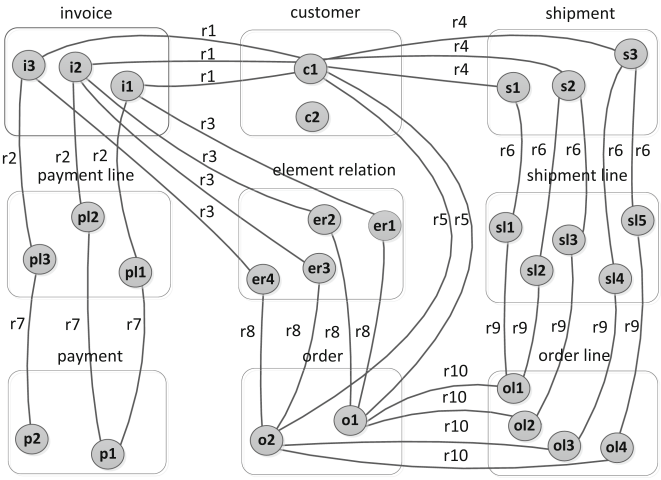


Fig. 2. An example of an object model.

An object model consists of objects, object relations and two functions indicating the class and attributes of each object. In the database context, an object model represents the state of a database at some moment, where objects correspond to records in tables and relations correspond to dependencies between records.

Figure 2 shows an object model $OM = (Obj, Rel, class, objectAttr)$ which represents the state of the database indicated in Fig. 1. The 28 objects are depicted as grey dots. There are two objects *o1* and *o2* belonging to class "order", i.e., $class(o1) = class(o2) = order$. There are three object relations corresponding to class relationship *r1*, i.e., $(r1, c1, i1) \in Rel$, $(r1, c1, i2) \in Rel$ and $(r1, c1, i3) \in Rel$. The object $i1$ has an attribute named "creation_date" with a value "2017-08-15 09:13:27", i.e., $objectAttr(i1)(\text{"}creation\_date\text{"}) =$ "2017-08-15 09:13:27". Note that, the attributes of objects are not shown in the graph.

2.2 Database Changes

Unlike process-aware information systems (e.g., BPM/WFM systems), object-centric information systems keep transactions in "regular tables" in databases. A record in a table indicates the current state of an instance. It is difficult to reveal all the events impacting this record, e.g., the first record in "order_line" table tells nothing about events.

Fortunately, database technology often provides so-called *redo logs* (as shown in Table 1) that record the history of database changes. Similarly, most SAP systems provide two *change tables*, i.e., CDHDR (recording all database changes) and CDPOS (recording details of each change in CDHDR) as shown in Table 2. If redo logs and change tables are not available, then domain knowledge is needed to obtain changes, i.e., timestamps/dates in particular columns refer to the creation

of the row or to changes of values. Using domain knowledge, these timestamps can reconstruct database changes.

Table 1. A fragment of a redo log in which each line corresponds to a change in the database.

#	Time	Redo
1	2017-08-11 10:33:37	Insert into "order" ("id", "creation_date", "customer") values ("o1", "2017-08-11 10:33:37", "c1")
2	2017-08-11 10:33:37	Insert into "order_line" ("id", "order", "product", "quantity", "price") values ("ol1", "o1", "computer", "2", "1190")
3	2017-08-11 10:33:37	Insert into "order_line" ("id", "order", "product", "quantity", "price") values ("ol2", "o1", "phone", "3", "476")

Table 2. A fragment of the CDHDR table (left) and CDPOS table (right).

HeadID	Date	Time	Op
100	2017-08-11	10:33:37	I
101	2017-08-11	10:33:37	I
102	2017-08-11	10:33:37	I
103	2017-08-13	16:28:15	I
104	2017-08-14	11:36:35	I

ItemID	HeadID	TabName	Key	FieldName	Op	NewValue	OldValue
1	100	order	o1	id	I	o1	none
2	100	order	o1	creation_date	I	2017-08-11 10:33:37	none
3	100	order	o1	customer	I	c1	none
4	101	order_line	ol1	id	I	ol1	none
5	101	order_line	ol1	order	I	o1	none

Each change corresponds to one execution of an SQL sentence, i.e., one event on the database level. In this paper, the extracted logs cater to people who operate information systems and may be unfamiliar with databases. Therefore, the extracted events are on the information system level rather than the database level. In other words, an example of an event is a "create order" operation rather than an execution of "insert an order record", which makes extracted logs more intuitive and understandable. Note that, an event may correspond to multiple changes, e.g., the three changes in Table 1 correspond to one "create order" event, and these changes may have the same timestamp (i.e., it is not necessary that each change has a unique timestamp). In real applications, the derived changes may cover multiple processes. For a given process, we can scope the changes using tricks such as based on a time window or classes involved [13].

2.3 eXtensible Object-Centric (XOC) Log Format

Based on database changes and the current state of a database, it is possible to restore all previous states of the database. The idea of the XOC format is that one event represents one operation on the information system and corresponds to an object model providing a *snapshot of the system just after this operation*. Besides, each event has an event type to indicate the executed activity (e.g., "create order"), and refers to some objects modified by the event.

Table 3. A fragment of the XOC log extracted from the motivating example data in Fig. 1.

Index	Event	Event type	References	Object model	
				Objects	Relations
1	*co1*	*create order* (*co*)	{*o1*, *ol1*, *ol2*}	{*c1*, *c2*, *o1*, *ol1*, *ol2*}	{(*r5*, *c1*, *o1*), (*r10*, *o1*, *ol1*), (*r10*, *o1*, *ol2*)}
2	*co2*	*create order* (*co*)	{*o2*, *ol3*, *ol4*}	{*c1*, *c2*, *o1*, *ol1*, *ol2*, *o2*, *ol3*, *ol4*}	{(*r5*, *c1*, *o1*), (*r10*, *o1*, *ol1*), (*r10*, *o1*, *ol2*), (*r5*, *c1*, *o2*), (*r10*, *o2*, *ol3*), (*r10*, *o2*, *ol4*)}
3	*cs1*	*create shipment* (*cs*)	{*s1*, *sl1*}	{*c1*, *c2*, *o1*, *ol1*, *ol2*, *o2*, *ol3*, *ol4*, *s1*, *sl1*}	{(*r5*, *c1*, *o1*), (*r10*, *o1*, *ol1*), (*r10*, *o1*, *ol2*), (*r5*, *c1*, *o2*), (*r10*, *o2*, *ol3*), (*r10*, *o2*, *ol4*), (*r4*, *c1*, *s1*), (*r6*, *s1*, *sl1*), (*r9*, *ol1*, *sl1*)}
4	*ci1*	*create invoice* (*ci*)	{*i1*, *er1*}	{*c1*, *c2*, *o1*, *ol1*, *ol2*, *o2*, *ol3*, *ol4*, *s1*, *sl1*, *i1*, *er1*}	{(*r5*, *c1*, *o1*), (*r10*, *o1*, *ol1*), (*r10*, *o1*, *ol2*), (*r5*, *c1*, *o2*), (*r10*, *o2*, *ol3*), (*r10*, *o2*, *ol4*), (*r4*, *c1*, *s1*), (*r6*, *s1*, *sl1*), (*r9*, *ol1*, *sl1*), (*r1*, *c1*, *i1*), (*r3*, *i1*, *er1*), (*r8*, *o1*, *er1*)}
5	*cs2*	*create shipment* (*cs*)	{*s2*, *sl2*, *sl3*}	{*c1*, *c2*, *o1*, *ol1*, *ol2*, *o2*, *ol3*, *ol4*, *s1*, *sl1*, *i1*, *er1*, *s2*, *sl2*, *sl3*}	{(*r5*, *c1*, *o1*), (*r10*, *o1*, *ol1*), (*r10*, *o1*, *ol2*), (*r5*, *c1*, *o2*), (*r10*, *o2*, *ol3*), (*r10*, *o2*, *ol4*), (*r4*, *c1*, *s1*), (*r6*, *s1*, *sl1*), (*r9*, *ol1*, *sl1*), (*r1*, *c1*, *i1*), (*r3*, *i1*, *er1*), (*r8*, *o1*, *er1*), (*r4*, *c1*, *s2*), (*r6*, *s2*, *sl2*), (*r9*, *ol1*, *sl2*), (*r6*, *s2*, *sl3*), (*r9*, *ol2*, *sl3*)}

Definition 2 (eXtensible Object-Centric Event Log). *Let* $\mathcal{U}_E$ *be the universe of* events *and* $\mathcal{U}_{ET}$ *be the universe of* event types. *An eXtensible Object-Centric (XOC)* event log *is a tuple* $L = (E, act, refer, om, \prec)$, *where*

- $E \subseteq \mathcal{U}_E$ *is a set of events,*
- $act \in E \rightarrow \mathcal{U}_{ET}$ *maps events onto event types,*
- $refer \in E \rightarrow \mathcal{P}(\mathcal{U}_O)$ *relates events to sets of objects,*
- $om \in E \rightarrow \mathcal{U}_{OM}$ *maps an event to the object model just after the event occurred,*
- $\prec \subseteq E \times E$ *defines a total order on events.*

$\mathcal{U}_L$ *is the universe of XOC event logs.*

Table 3 shows a XOC log example. The "Event" column specifies the set of events while the "Index" column indicates the total order on events. The last four columns show the corresponding event type, object references (i.e., modified objects) and object model (i.e., objects and relations) of each event, respectively. Note that, the information of objects (e.g., attribute values) is left out in Table 3. In some cases, object models may increase dramatically since an object model contains unchanged contents of the previous one and new contents. This problem can be solved by some storage tricks, e.g., only storing updated information, which falls out of the scope of this paper.

3 Extracting Object-Centric Event Logs from Databases

In this section, we propose an approach to extract XOC logs from databases based on object-centric data (i.e., database tables) and database changes.

3.1 Formalizations of Object-Centric Data

This subsection provides a formal definition to describe object-centric data (i.e., database tables), which refers to the notations in [5,13].

Definition 3 (Data Model). *Let $V \subseteq \mathcal{U}_{Val}$ be a set of values. A data model is a tuple $DM = (C, A, classAttr, val, PK, FK, classPK, classFK, keyRel, keyAttr, refAttr)$ such that*

- *$C \subseteq \mathcal{U}_C$ is a set of classes,*
- *$A \subseteq \mathcal{U}_{Attr}$ is a set of attribute names,*
- *$classAttr \in C \rightarrow \mathcal{P}(A)$ maps each class onto a set of attribute names,*[4]
- *$val \in A \rightarrow \mathcal{P}(V)$ maps each attribute onto a set of values,*
- *PK is a set of primary keys and FK is a set of foreign keys, where PK and FK are assumed to be disjoint in this paper (i.e., $PK \cap FK = \emptyset$) and $K = PK \cup FK$,*
- *$classPK \in C \rightarrow PK$ maps each class onto a primary key,*
- *$classFK \in C \rightarrow \mathcal{P}(FK)$ maps each class onto a set of foreign keys,*
- *$keyRel \in FK \rightarrow PK$ maps each foreign key onto a primary key,*
- *$keyAttr \in K \rightarrow \mathcal{P}(A)$ maps each key onto a set of attributes, and*
- *$refAttr \in FK \times A \not\rightarrow A$ maps each pair of a foreign key and an attribute onto an attribute from the corresponding primary key. That is, $\forall k \in FK : \forall a, a' \in keyAttr(k) : (refAttr(k, a) \in keyAttr(keyRel(k)) \wedge (refAttr(k, a) = refAttr(k, a') \Longrightarrow a = a')$.*

$\mathcal{U}_{DM}$ is the universe of data models.

A *data model* describes the structure of the object-centric data. More precisely, a class represents a table while an attribute represents a column in a table. Function *classAttr* specifies the attributes of a class and function *val* indicates possible values for an attribute. *classPK* and *classFK* define the primary key (PK) and foreign keys (FKs) for each table, respectively. Given a FK, *keyRel* indicates its referred PK.

Take the tables in Fig. 1 as an example. $C = \{p, o, s, er, ol, sl, pl, i, c\}$ and $A = \{id, creation\_date, amount, \ldots, address\}$. $classAttr(p) = \{id, creation\_date, amount\}$ and $val(amount) = \mathbb{N}$ where $p \in C$ (i.e., the "payment" table).

Definition 4 (Records). *Let $DM = (C, A, classAttr, val, PK, FK, classPK, classFK, keyRel, keyAttr, refAttr)$ be a data model and $M^{DM} = \{map \in A \not\rightarrow V \mid \forall a \in dom(map) : map(a) \in val(a)\}$ be the set of mappings of DM. $R^{DM} = \{(c, map) \in C \times M^{DM} \mid dom(map) = classAttr(c)\}$ is the set of all records of DM if $\forall (c, map), (c, map') \in R^{DM} : (\forall a \in keyAttr(classPK(c)) : map(a) = map'(a)) \Rightarrow map = map'$.*

Given a data model, Definition 4 formalizes the possible records of the model. For instance, the first record in the "payment_line" table can be formalized as (c, map) where $c = pl$, $dom(map) = \{id, payment, invoice, amount\}$, and $map(id) = pl1$, $map(payment) = p1$, $map(invoice) = i1$, $map(amount) = 2380$.

[4] $\mathcal{P}(X)$ is the power set of X, i.e., $Y \in \mathcal{P}(X)$ if $Y \subseteq X$.

Definition 5 (Data Set). *An object-centric data set is a tuple* $DS = (DM, RS)$ *where* $DM \in \mathcal{U}_{DM}$ *is a data model and* $RS \subseteq R^{DM}$ *is a set of records of* DM. $\mathcal{U}_{DS}$ *is the universe of data sets.*

3.2 Formalizations of Database Changes

After formalizing the data perspective (i.e., data model and records), this subsection formalizes the changes (i.e., adding, updating, or deleting records) in databases.

Definition 6 (Change Types). *Let* $DM = (C, A, classAttr, val, PK, FK, classPK, classFK, keyRel, keyAttr, refAttr)$ *be a data model.* $CT^{DM} = CT^{DM}_{add} \cup CT^{DM}_{upd} \cup CT^{DM}_{del}$ *is the set of change types composed of the following pairwise disjoint sets:*

- $CT^{DM}_{add} = \{(\oplus, c, A') \mid c \in C \wedge A' \subseteq classAttr(c) \wedge keyAttr(classPK(c)) \subseteq A'\}$ *are the change types for adding records of* DM,
- $CT^{DM}_{upd} = \{(\oslash, c, A') \mid c \in C \wedge A' \subseteq classAttr(c) \wedge keyAttr(classPK(c)) \cap A' = \emptyset\}$ *are the change types for updating records of* DM, *and*
- $CT^{DM}_{del} = \{(\ominus, c, A') \mid c \in C \wedge A' = classAttr(c)\}$ *are the change types for deleting records of* DM.

$\mathcal{U}_{CT}$ *is the universe of change types.*

Definition 7 (Changes). *Let* DM *be a data model,* R^{DM} *be the set of records of* DM, $CT^{DM} = CT^{DM}_{add} \cup CT^{DM}_{upd} \cup CT^{DM}_{del}$ *be the set of change types and* $map_{null} \in \emptyset \rightarrow V$ *be a function with the empty set as domain.* $CH^{DM} = CH^{DM}_{add} \cup CH^{DM}_{upd} \cup CH^{DM}_{del}$ *is the set of changes composed of the following pairwise disjoint sets:*

- $CH^{DM}_{add} = \{(\oplus, c, A', map_{old}, map_{new}) \mid (\oplus, c, A') \in CT^{DM}_{add} \wedge map_{old} = map_{null} \wedge (c, map_{new}) \in R^{DM}\}$,
- $CH^{DM}_{upd} = \{(\oslash, c, A', map_{old}, map_{new}) \mid (\oslash, c, A') \in CT^{DM}_{upd} \wedge (c, map_{old}) \in R^{DM} \wedge (c, map_{new}) \in R^{DM}\}$, *and*
- $CH^{DM}_{del} = \{(\ominus, c, A', map_{old}, map_{new}) \mid (\ominus, c, A') \in CT^{DM}_{del} \wedge (c, map_{old}) \in R^{DM} \wedge map_{new} = map_{null}\}$.

A *change* $(op, c, A', map_{old}, map_{new})$ corresponds to an SQL sentence, i.e., adding, updating or deleting a record in a table. Its *change type* (op, c, A') indicates which table (i.e., c) the record is in, which columns (i.e., A') of the record are impacted and how the record is impacted (indicated by op, i.e., adding, updating or deleting), while its old and new mappings (i.e., map_{old} and map_{new}) specify the contents of the record before and after the change, respectively. Note that, A' provides more power to produce different change types, i.e., changes impacting different attributes in the same table can be clarified into different change types, which is not considered in [5].

Definition 8 (Change Occurrence, Change Log). *Let DM be a data model and CH^{DM} be the set of changes of DM. Let TS be the universe of timestamps. $CO^{DM} = CH^{DM} \times TS$ is the set of all possible change occurrences of DM. A change log $CL = \langle co_1, co_2, \ldots, co_n \rangle \in (CO^{DM})^*$ is a sequence of change occurrences such that time is non-decreasing, i.e., $ts_i \leqslant ts_j$ for any $co_i = (ch_i, ts_i)$ and $co_j = (ch_j, ts_j)$ with $1 \leqslant i < j \leqslant n$. $\mathcal{U}_{CL}$ is the universe of change logs.*

A *change occurrence* $co = (ch, ts)$ represents a change ch happened at ts. It corresponds to one row in the redo log or CDHDR table. A *change log* consists of a list of change occurrences which are sorted by timestamps such that time is non-decreasing. Note that, *change logs* record behavior on the database level (e.g., an execution of "insert an order record") while XOC logs record behavior on the information system level (e.g., a "create order" operation) (cf. Sect. 2.2).

Definition 9 (Effect of a Change). *Let DM be a data model. CH^{DM} is the set of changes of DM and $ch = (op, c, A', map_{old}, map_{new}) \in CH^{DM}$ is a change. $DS_{old} = (DM, RS_{old})$ and $DS_{new} = (DM, RS_{new})$ are two data sets. DS_{new} is generated after the change ch on DS_{old} and denote $DS_{old} \overset{ch}{\rightarrow} DS_{new}$ if and only if*

- $RS_{new} = \{(c', map') \in RS_{old} \mid (c', map') \neq (c, map_{old})\} \cup \{(c, map_{new}) \mid op \neq \ominus\}$ *or*
- $RS_{old} = \{(c', map') \in RS_{new} \mid (c', map') \neq (c, map_{new})\} \cup \{(c, map_{old}) \mid op \neq \oplus\}$.

Definition 10 (Effect of a Change Log). *Let DM be a data model and $CL = \langle co_1, co_2, \ldots, co_n \rangle \in \mathcal{U}_{CL}$ be a change log. There exist data sets DS_0, $DS_1, \ldots$, $DS_n \in \mathcal{U}_{DS}$ such that $DS_0 \overset{co_1}{\rightarrow} DS_1 \overset{co_2}{\rightarrow} DS_2 \ldots \overset{co_n}{\rightarrow} DS_n$. Hence, change log CL results in data set DS_n when starting in DS_0. This is denoted by $DS_0 \overset{CL}{\rightarrow} DS_n$.*

Given a data set $DS_{old} = (DM, RS_{old})$, a change results in a new data set DS_{new} through adding, updating or deleting a record in the record set RS_{old} of DS_{old} (cf. Definition 9). Similarly, a change log results in a data set DS_n through orderly accumulating the effects of all changes on the initial data set DS_0, indicated by Definition 10.

3.3 Formalizations of Extracting Logs

A XOC log $(E, act, refer, om, \prec)$ consists of events, event types, object references, object models and event ordering. For extracting events, one needs to specify some event types based on domain knowledge which indicates the relations between event types (classifying behavior at the information system level) and change types (classifying behavior at the database level) (cf. Sect. 2.2). An example of the knowledge is that the "create order" event type consists of two change types, i.e., "insert one order record" and "insert one or more order line records", since a click on the "create order" button inserts one record in the "order" table and one or more records in the "order_line" table.

Definition 11 (Cardinalities). *$\mathcal{U}_{Card} = \{X \in \mathcal{P}(\mathbb{N})\backslash\{\emptyset\}\}$ defines the universe of all cardinalities. A cardinality (an element of $\mathcal{U}_{Card}$) specifies a non-empty set of integers.*

Definition 12 (Event Types). *Let DM be a data model and CT^{DM} be the set of change types of DM. $ET^{DM} = \{et \in \mathcal{P}(\mathcal{U}_{Card} \times CT^{DM})\backslash\{\emptyset\} \mid \forall(card_1, ct_1), (card_2, ct_2) \in et : (ct_1 = ct_2 \Rightarrow card_1 = card_2)\}$ is the set of event types of DM.*

An *event type* is defined as a set of tuples of a *cardinality* and a change type, where the cardinality describes the quantitative relation between the event type and the change type. For example, the "create order" event type is denoted as $\{(\{1\}, (\oplus, o, A'_1)), (\{1..*\}, (\oplus, ol, A'_2))\}$ which means a "create order" event adds precisely one record in the "order" (o) table, and at least one record in the "order_line" (ol) table. Definition 12 specifies a concrete realization for event types while $\mathcal{U}_{ET}$ (cf. Definition 2) only abstractly defines the universe.

Definition 13 (Events). *Let DM be a data model, CO^{DM} be the set of change occurrences of DM and ET^{DM} be the set of event types of DM. $E^{DM} = \{e \in (CO^{DM})^*\backslash\{\emptyset\} \mid \forall(ch_i, ts_i), (ch_j, ts_j) \in e : (i < j \Rightarrow ts_i \leqslant ts_j)\}$ is the set of events of DM. Function $possibleE \in ET^{DM} \rightarrow \mathcal{P}(E^{DM})$ returns possible events of an event type such that $possibleE(et) = \{e \in E^{DM} \mid \forall((op, c, A', map_{old}, map_{new}), ts) \in e : (\exists card \in \mathcal{U}_{card} : (card, (op, c, A')) \in et) \wedge \forall(card', (op', c', A'')) \in et : |\{((op', c', A'', map'_{old}, map'_{new}), ts') \in e\}| \in card'\}$.*

An event is a non-empty sequence of change occurrences such that time is non-decreasing. Function *possibleE* gives all possible events corresponding to one event type. For instance, $e = \langle((\oplus, o, A, map_{old}, map_{new}), ts), ((\oplus, ol, A', map'_{old}, map'_{new}), ts)\rangle$ is a possible event of the event type "create order" (co), i.e., $e \in possibleE(co)$. Definition 13 specifies a concrete realization for events.

Definition 14 (Extracting Event Types). *Let DM be a data model. $E \subseteq E^{DM}$ is a set of events. $ET \subseteq ET^{DM}$ is a predefined set of event types where $\forall et_1, et_2 \in ET : possibleE(et_1) \cap possibleE(et_2) = \emptyset$. Function $extractET \in E \rightarrow ET$ maps an event to an event type such that $extractET(e) = et$ where $e \in possibleE(et)$.*

Given a predefined set of event types whose possible events are disjoint, function *extractET* maps an event to an event type in the set. In this paper, we assume there exists exactly one possible type in the predefined set for the event.

Definition 15 (Mapping Record into Object). *Let $DM = (C, A, classAttr, val, PK, FK, classPK, classFK, keyRel, keyAttr, refAttr)$ be a data model and R^{DM} be the set of records of DM. Function $extractO \in R^{DM} \rightarrow \mathcal{U}_O$ maps a record (c, map) to an object such that $extractO((c, map)) = (c, map_k) \in C \times M^{DM}$ where*

- $dom(map_k) = keyAttr(classPK(c))$, *and*
- $\forall a \in dom(map_k) : map_k(a) = map(a)$.

Function *extractO* filters in the mapping for attributes corresponding to the primary key of a record, resulting in an object. This function specifies a concrete realization, i.e., a tuple (c, map_k), for each object in $\mathcal{U}_O$ (cf. Definition 1).

Definition 16 (Mapping Data Set into Object Model). *Let* $DS = (DM, RS)$ *be a data set where* $DM = (C, A, classAttr, val, PK, FK, classPK, classFK, keyRel, keyAttr, refAttr)$ *is a data model and RS is a set of records. Function* $extractOM \in \mathcal{U}_{DS} \rightarrow \mathcal{U}_{OM}$ *maps a data set to an object model such that* $extractOM(DS) = (Obj, Rel, class, objectAttr)$ *where*

- $Obj = \{extractO(r) \mid r \in RS\}$,
- $Rel = \{(rt, extractO((c, map)), extractO((c', map')) \mid rt = (c, pk, c', fk) \in C \times PK \times C \times FK \wedge (c, map) \in RS \wedge (c', map') \in RS \wedge pk = classPK(c) \wedge fk \in classFK(c') \wedge keyRel(fk) = pk \wedge \forall a \in keyAttr(fk) : map'(a) = map(refAttr(fk, a))\}$,
- $\forall (c, map_k) \in Obj : class((c, map_k)) = c$, *and*
- $\forall (c, map_k) \in Obj : objectAttr((c, map_k)) = map$ *where* $(c, map) \in RS$ *and* $extractO((c, map)) = (c, map_k)$.

Function *extractOM* maps a data set into an object model. More precisely, if each attribute value corresponding to a foreign key of a record r is equal to the value of the attribute identified by function *refAttr* in another record r', there exists an object relation between $extractO(r)$ and $extractO(r')$.

Definition 17 (Transforming Change Log Into Event Sequence). *Function* $extractES \in \mathcal{U}_{CL} \rightarrow (\mathcal{U}_E)^*$ *extracts an event sequence from a change log such that* $\forall CL = \langle co_1, co_2, \ldots, co_n \rangle \in \mathcal{U}_{CL} : extractES(CL) = \langle e_1, e_2, \ldots, e_m \rangle$ *where*

- $\{co \in CL\} = \{co' \mid co' \in e_i \wedge e_i \in \langle e_1, e_2, \ldots, e_m \rangle\}$, *and*
- $\forall e_i, e_j \in \langle e_1, e_2, \ldots, e_m \rangle : (\forall (ch, ts) \in e_i, (ch', ts') \in e_j : i < j \Rightarrow ts < ts')$.

A reverse function $restoreES \in (\mathcal{U}_E)^* \rightarrow \mathcal{U}_{CL}$ *which restores an event sequence to a change log such that* $restoreES(\langle e_1, e_2, \ldots, e_m \rangle) = \langle co_1, co_2, \ldots, co_n \rangle$.

Function *extractES* transforms a change log (i.e., a change occurrence sequence) into an event sequence by grouping change occurrences at the *same time* into an event without modifying the order of these change occurrences. Note that, it is possible to group change occurrences based on *domain knowledge* instead of *time*, which provides more freedom for extracting events (e.g., overlapping events). On the contrary, function *restoreES* transforms an event sequence into a change occurrence sequence, i.e., $restoreES(\langle e_1, e_2, \ldots, e_m \rangle) = \langle co_1, co_2, \ldots, co_n \rangle$ if and only if $extractES(\langle co_1, co_2, \ldots, co_n \rangle) = \langle e_1, e_2, ..., e_m \rangle$.

Definition 18 (Extracting XOC Event Log). *Function* $extractL \in \mathcal{U}_{DS} \times \mathcal{U}_{CL} \rightarrow \mathcal{U}_L$ *extracts a XOC event log from a data set and a change log such that* $\forall DS \in \mathcal{U}_{DS}, CL \in \mathcal{U}_{CL} : extractL(DS, CL) = (E, act, refer, om, \prec)$ *where*

- $E = \{e \in \langle e_1, e_2, \ldots, e_m \rangle\}$ *where* $\langle e_1, e_2, \ldots, e_m \rangle = extractES(CL)$,
- $act = extractET$,
- $\forall e_i \in E : refer(e_i) = \{o \mid \exists((op, c, A', map_{old}, map_{new}), ts) \in e_i : o = extractO(c, map_{new}) \wedge op \neq \ominus\} \cup \{o \mid \exists((\ominus, c, A', map_{old}, map_{new}), ts) \in e_i : o = extractO(c, map_{old})\}$,
- $\forall e_i \in E : om(e_i) = OM_i$ *such that* $OM_i \overset{CL'}{\rightarrow} extractOM(DS)$ *where* $CL' = restoreES(tl^{m-i}(\langle e_1, e_2, \ldots, e_m \rangle))$ *and* $\langle e_1, e_2, \ldots, e_m \rangle = extractES(CL)$,[5] *and*
- $\prec = \{(e_i, e_j) \mid e_i \in E \wedge e_j \in tl^{m-i}(\langle e_1, e_2, \ldots, e_m \rangle)$ *where* $\langle e_1, e_2, \ldots, e_m \rangle = extractES(CL)$.

Function *refer* is obtained through identifying the impacted (added, updated or deleted) objects by each event. The data set indicates the object model (i.e., $extractOM(DS)$) corresponding to the last event (i.e., e_m). In order to obtain the object model corresponding to one event e_i, we get (i) its suffix event sequence $tl^{m-i}(\langle e_1, e_2, \ldots, e_m \rangle)$ (i.e., $\langle e_{i+1}, e_{i+2}, \ldots, e_m \rangle$), (ii) the corresponding change sequence CL' through *extractES* and (iii) the object model OM_i through $\rightarrow$.

4 Implementation

Our approach has been implemented in the "XOC Log Generator" Plugin in *ProM 6 Nightly builds*, which takes tables (from a database or csv files) as input and automatically generates a XOC log. For instance, taking the motivating data as input, this plugin generates an event log with 10 events, the first five ones of which are shown in Table 3.[6]

A XOC log reveals the evolution of a database along with the events from its corresponding information system. As shown in Fig. 3, after the occurrence of an event (denoted by a black dot), its corresponding object model (denoted by a cylinder) is updated by adding, updating or deleting objects (highlighted in red). Besides, XOC logs provide methods to correlate events by objects. For instance, two events are correlated if they refer to (indicated by dotted lines) two related objects, e.g., $co1$ and $cs1$ are correlated since $co1$ refers to $ol1$, $cs1$ refers to $sl1$ and $ol1$ is related to $sl1$.

Figure 4 shows the comparison between the XES log and the XOC log derived from the motivating data in Fig. 1. More precisely, Fig. 4(a) reveals the behavioral perspective of the XOC log after correlating events and Fig. 4(b) shows two cases $o1$ and $o2$. Our log is able to represent one-to-many and many-to-many relations by removing the case notion, while the XES log splits events based on

[5] $tl^k(\sigma)$ means to get the last k elements of a sequence σ.

[6] The involved data and more real life examples can be found at http://www.win.tue.nl/ocbc/.

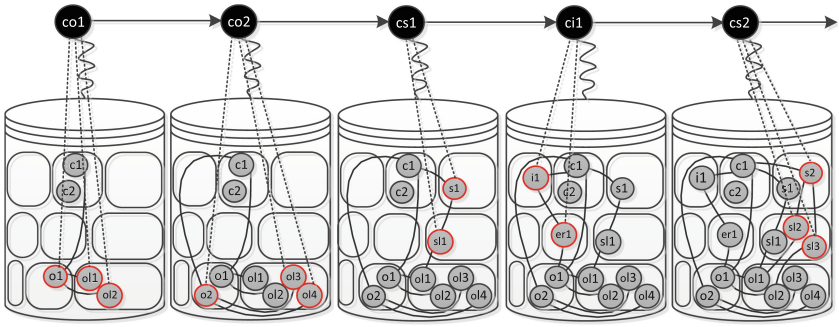


Fig. 3. A XOC log (in Table 3) revealing the evolution of a database. (Color figure online)

a straightjacket case id. The split performed in the XES log results in two separated cases without interactions in between as well as leading to convergence (e.g., *cp*1 appears in two cases) and divergence (e.g., multiple instances of the activity *cp* appear in *o*2) problems.

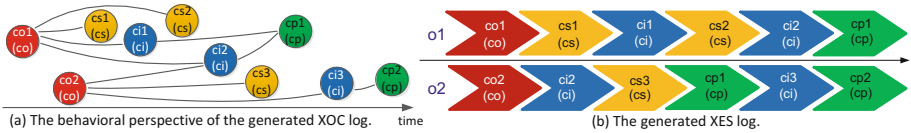


Fig. 4. A comparison which indicates XOC logs can avoid data convergence and divergence.

Besides the advantages over the existing log format, XOC logs open a door for other techniques. Based on XOC logs, we can discover *object-centric behavioral constraint (OCBC)* models [8], as shown in Fig. 5. Figure 6 shows parts of three example models discovered from the generated XES log in Fig. 4(b), which displays the problems caused by data convergence and divergence. In (a) we see that the negative relation denoted by *d*5 with a dotted line cannot be discovered in the declare model, since in case *o*2, the "create payment" event *cp*1 is wrongly related to its subsequent "create invoice" event *ci*3. The discovered directly follow graph (DFG) in (b) indicates that "create invoice" happens 4 times, violating the real frequency of 3 times due to the duplication of *ci*2. In (c) we observe that, since the multiple instance of activities "create invoice" and "create payment" cannot be distinguished, two loops with two implicit transitions are discovered, resulting in an imprecise Petri net. In contrast, since XOC logs are not affected by convergence and divergence, the discovered OCBC model (cf. Fig. 5) can show the correct frequencies (i.e., numbers in red) and the

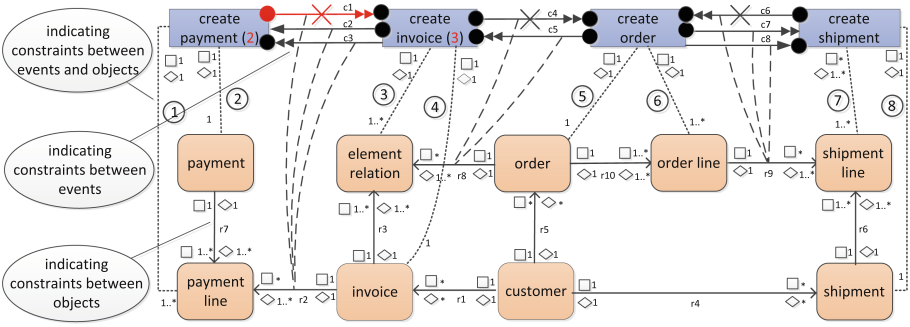


Fig. 5. The OCBC model discovered from the generated XOC log. (Color figure online)

missing negative relation (i.e., $c1$)[7], avoiding the problems confronted by these three models. Besides, compared with existing modeling notations, such as the models in Fig. 6, OCBC models can show the data perspective, the behavioral perspective and the interactions in between in a single diagram. Moreover, since XOC logs can correlate events by objects and have a richer data perspective, they can be used to *check conformance* for more implicit deviations by considering multiple instances and data objects [14]. Additionally,since the discovered OCBC model indicates constraints in/between objects and events, it is possible to *predict* future events and objects based on observed ones.

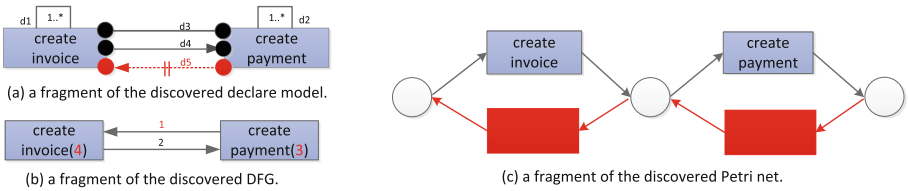


Fig. 6. Problems suffered by three kinds of models discovered from the generated XES log. (Color figure online)

5 Related Work

In this section, we review the existing event log formats and introduce some researches which extract event logs from databases.

Event Log Formats. Event logs serve as the input for many process mining techniques. The XES log format [1] which stands for *eXtensible Event Stream* (www.xes-standard.org) is the de facto exchange format for process mining. This

[7] The notation for the negative relation in OCBC models is different from that in declare models.

format is supported by tools such as ProM and implemented by OpenXES, an open source java library for reading, storing and writing XES logs. The XES format cannot deal well with object-centric data. [4] proposes a meta model to abstract the object-centric data and redo logs into different types of entities such as attributes, objects, relations and events. The meta model covers both behavioral and data perspectives and builds a bridge to connect databases with process mining. This meta model is not a concrete log format, but transforms the object-centric data into "bricks" which can constitute logs to enable process mining techniques. Our XOC log format combines the "bricks" from [4] in an object-centric way, i.e., it focuses more on the data perspective unlike the XES format. Objects are replacing a case notion to correlate events, which makes XOC logs able to deal with one-to-many and many-to-many relations. Moreover, through defining object models, a XOC log reveals the evolution of a database through time.

Extracting Event Logs. In order to obtain event logs, researches propose a number of techniques and tools. A popular tool is XESame[8], which is used to convert databases into XES event logs. [13] conceptualizes a database view over event data based on the idea that events leave footprints by changing the underlying database, which are captured by redo logs of database systems. Triggered by this idea, [5] proposes a method to extract XES logs from databases by identifying a specific trace id pattern. [3] proposes an approach to flexibly extract XES event logs from relational databases, by leveraging the ontology-based data access paradigm and [2] provides a tool (*onprom*) to extract event logs from legacy data based on an ontology. In order to obtain logs from non-process-aware information systems, [11] correlates events into process instances using similarity of their attributes. [7] provides an overview of the decisions that impact the quality of the event logs constructed from database data. [6] proposes an analysis system which allows users to define events, resources and their inter-relations to extract logs from SAP transactions. [12] addresses merging logs produced by disintegrated systems that cover parts of the same process by choosing a "main" process. Artifact-centric approaches [9,10] try to extract artifacts (i.e., business entities) and address the possibility of many-to-many relationships between artifacts. Compared with the existing approaches, our approach outputs object-centric logs, i.e., XOC logs, rather than process-centric logs, i.e., XES logs. The main advantage of object-centric logs is the new kinds of analyses that they enable. Data-aware process model discovery [8], and new conformance checking techniques [14] that exploit data relations are examples of approaches directly applicable to XOC logs. Another important contribution is that our approach supports the abstraction from low level database events (like the ones obtained in [5]) to high level events (e.g., from the original information system).

[8] http://www.processmining.org/xesame/start.

6 Conclusion

In this paper we proposed an approach to extract event logs from object-centric data (i.e., database tables), using the eXtensible Object-Centric (XOC) log format. The XOC format provides a process mining view on databases. Compared with existing log formats, such as XES, it has the following advantages:

- By removing the case notion and correlating events with objects, XOC logs can perfectly deal with one-to-many and many-to-many relations, avoiding convergence and divergence problems and displaying interactions between different instances.
- An object in XOC logs contains as much information as its corresponding record in the database. By extending the data perspective, XOC logs retain the data quality.
- The object model of an event represents a snapshot of the database just after the event occurrence. Based on this idea, the log provides a view of the evolution of the database, along with the operations which triggered changes in the database.

Besides, XOC logs serve as a starting point for a new line of future techniques. Based on experiments implemented on the generated XOC logs, it is possible to discover OCBC models (cf. Fig. 5) to describe the underlying process in an object-centric manner [8]. Additionally, taking a XOC log and an OCBC model as input, many deviations which cannot be detected by existing approaches, can be revealed by new conformance checking techniques [14]. Moreover, prediction of future events and objects is also enabled according to the constraints indicated by the discovered OCBC model. Note that, the OCBC discovery, conformance checking and prediction are just examples of potential applications of XOC logs. It is possible to propose more techniques based on XOC logs, e.g., discovering other types of data-aware process models.

References

1. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. IEEE Std. 1849-2016, pp. i–48 (2016)
2. Calvanese, D., Kalayci, T.E., Montali, M., Tinella, S.: Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 220–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_16
3. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 140–153. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_12
4. González López de Murillas, E., Reijers, H.A., van der Aalst, W.M.P.: Connecting databases with process mining: a meta model and toolset. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) BPMDS/EMMSAD -2016. LNBIP, vol. 248, pp. 231–249. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_15

5. González López de Murillas, E., van der Aalst, W.M.P., Reijers, H.A.: Process mining on databases: unearthing historical data from redo logs. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 367–385. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_25
6. Ingvaldsen, J.E., Gulla, J.A.: Preprocessing support for large scale process mining of SAP transactions. In: ter Hofstede, A., Benatallah, B., Paik, H.-Y. (eds.) BPM 2007. LNCS, vol. 4928, pp. 30–41. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78238-4_5
7. Jans, M., Soffer, P.: From relational database to event log: decisions with quality impact. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 588–599. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_46
8. Li, G., de Carvalho, R.M., van der Aalst, W.M.P.: Automatic discovery of object-centric behavioral constraint models. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 43–58. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_4
9. Lu, X., Nagelkerke, M., van de Wiel, D., Fahland, D.: Discovering interacting artifacts from ERP systems. IEEE Trans. Serv. Comput. **8**(6), 861–873 (2015)
10. Nigam, A., Caswell, N.S.: Business artifacts: an approach to operational specification. IBM Syst. J. **42**(3), 428–445 (2003)
11. Pérez-Castillo, R., et al.: Assessing event correlation in non-process-aware information systems. Softw. Syst. Model. **13**(3), 1117–1139 (2014)
12. Raichelson, L., Soffer, P.: Merging event logs with many to many relationships. In: Fournier, F., Mendling, J. (eds.) BPM 2014. LNBIP, vol. 202, pp. 330–341. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15895-2_28
13. van der Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. MP, pp. 105–128. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14430-6_8
14. van der Aalst, W.M.P., Li, G., Montali, M.: Object-centric behavioral constraints. CORR Technical report, arXiv.org e-Print archive (2017). https://arxiv.org/abs/1703.05740

Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development

Lidia López[1(✉)], Silverio Martínez-Fernández[2], Cristina Gómez[1], Michał Choraś[3,4], Rafał Kozik[3,4], Liliana Guzmán[2], Anna Maria Vollmer[2], Xavier Franch[1], and Andreas Jedlitschka[2]

[1] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{llopez, cristina, franch}@essi.upc.edu
[2] Fraunhofer IESE, Kaiserslautern, Germany
{silverio.martinez, liliana.guzman, anna-maria.vollmer, andreas.jedlitschka}@iese.fraunhofer.de
[3] ITTI Sp. Z O.O., Poznań, Poland
{mchoras, rafal.kozik}@itti.com.pl
[4] University of Science and Technology, UTP Bydgoszcz, Bydgoszcz, Poland

Abstract. Software quality is an essential competitive factor for the success of software companies today. Increasing the software quality levels of software products and services requires an adequate integration of quality requirements (QRs) in the software life-cycle, which is still scarcely supported in current rapid software development (RSD) approaches. One of the goals of the Q-Rapids (Quality-aware Rapid Software Development) method is providing tool support to decision-makers for QR management in RSD. The Q-Rapids method is based on gathering data from several and heterogeneous sources, to be aggregated into quality-related strategic indicators (e.g., customer satisfaction, product quality) and presented to decision-makers using a highly informative dashboard. The current release of Q-Rapids Tool provides four sets of functionality: (1) *data gathering* from source tools (e.g. GitLab, Jira, SonarQube, and Jenkins), (2) *aggregation of data* into three levels of abstraction (metrics, product/process factors, and strategic indicators), (3) *visualization of the aggregated data*, and (4) *navigation through the aggregated data.* The tool has been evaluated by four European companies that follow RSD processes.

Keywords: Agile · Decision-making · Quality requirement
Non-functional requirements · Rapid software development · Strategic indicator
Dashboard

1 Introduction

Software quality is an essential competitive factor for the success of software companies today. Increasing the software quality levels of software products and services requires an adequate integration of quality requirements (QRs) in the software life-cycle. However, QRs management is problematic in software development in general [1] and

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 200–208, 2018.
https://doi.org/10.1007/978-3-319-92901-9_17

in rapid software development (RSD) in particular [2]. In order to support decision-makers in QR management in RSD, the Q-Rapids (Quality-aware Rapid Software Development) method defines an evidence-based, data-driven quality-aware rapid software development approach in which QRs are incrementally elicited, refined and improved. Q-Rapids builds upon data gathered from several heterogeneous sources. Data is analysed and aggregated into quality-related strategic indicators (e.g., customer satisfaction, product quality) which are presented to decision-makers using a highly informative dashboard.

In this paper, we present the current status and first evaluation of the tool support for the Q-Rapids method, that we call Q-Rapids Tool. Nowadays, the tool gathers and aggregates data about system quality (e.g. SonarQube, Jenkins) and process productivity (e.g. GitLab, Jira) to visualize it from historical and current perspectives.

The rest of the paper is structured as follows. Section 2 briefly presents the Q-Rapids method. Section 3 introduces the architecture of the tool and describes each of its modules. Section 4 discusses the evaluation of the first release of the tool performed by the uses cases of the Q-Rapids project and Sect. 5 presents a roadmap for the following releases. Finally, Sect. 6 sketches some conclusions.

2 Q-Rapids Method

Q-Rapids is a data-driven, quality-aware rapid software development method that is being developed in the context of an EU H2020 project with the same name[1]. In Q-Rapids, quality requirements will be identified from available data and evaluated with respect to some selected indicators [3].

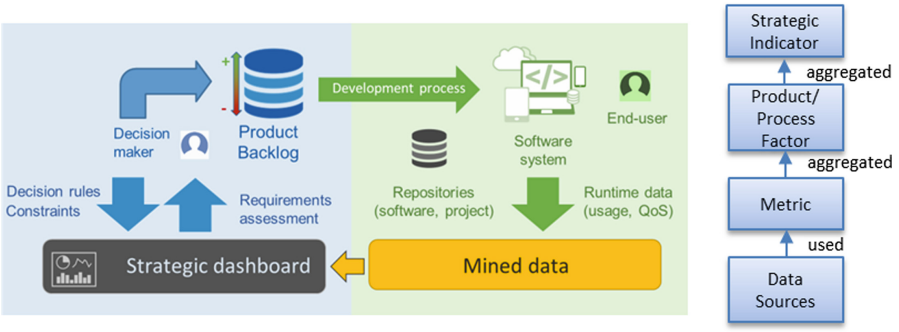


Fig. 1. (a) The Q-Rapids method and (b) quality model.

Q-Rapids aims to increase software quality through the following goals (see Fig. 1(a)):

[1] www.q-rapids.eu.

- Gathering and analyzing data from *project management tools*, *software repositories*, *quality of service* and *system usage*. The analysis of this data allows to assess systematically and continuously software quality using a set of quality-related indicators (e.g., customer satisfaction).
- Providing decision-makers with a highly informative dashboard to help them making data-driven, requirements-related strategic decisions in rapid cycles.
- Extending the rapid software development process considering the comprehensive integration of quality requirements and their management in a way that favors software quality and that brings a significant productivity increase to the software lifecycle.

In order to characterize quality-based strategic indicators, we define a quality model based on the Quamoco approach [4]. Quamoco faces the problem of traditional software quality models, which provide either abstract quality characteristics or concrete quality measurements, by integrating both aspects. The extra value of our quality model is to enable the aggregation from the raw data gathered to useful strategic indicators at the company level rendered in the dashboard. Concretely, metrics are computed from gathered data from data sources and are aggregated into product/process factors, and these factors are ultimately aggregated into strategic indicators (see Fig. 1(b)). The generic quality model, including the aggregations, used for the Q-Rapids Tool evaluation is reported in [5]. Concrete results of adopting Q-Rapids method, in one of the Q-Rapids project use cases, to characterize code quality are reported in [6].

One of the Q-Rapids project outcomes is a software tool to support the life-cycle development presented in Fig. 1(a) covering the first two project goals. The Q-Rapids Tool is being developed iteratively and its current version includes the following functionality:

- Gather information from several data sources.
- Aggregate the data from data sources to strategic indicators.
- Visualize the current assessment of the strategic indicators allowing decision-makers to analyze the current status of the project.
- Visualize historical data allowing decision-makers to make trend analysis to anticipate risks.
- Allow decision-makers to drill-down through different levels of data to understand the rationale of the current status.

3 Q-Rapids Tool

The architecture of the Q-Rapids Tool is depicted in Fig. 2. The components are grouped in two packages: *Data Gathering and Analysis* and *Strategic Decision Making*.

The *Data Gathering and Analysis* package includes three modules grouping the different phases of the data gathering and analysis process. The *Data Ingestion* module is the responsible of gathering the raw data from the different tools (*Data Producers*). Having this independent module helps us to integrate data from several data providers,

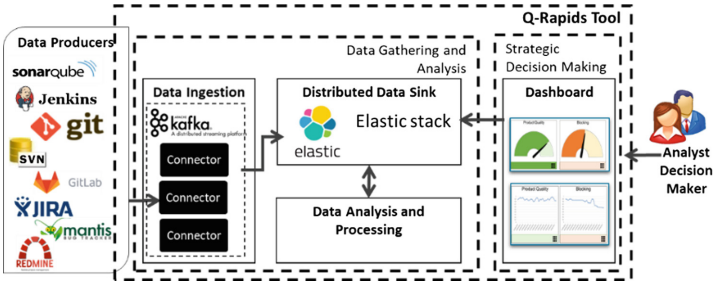


Fig. 2. Q-Rapids Tool architecture

making this heterogeneity of data transparent to the other modules. Once the data is in the system *(Distributed Data Sink)*, the *Data Analysis and Processing* module is responsible of executing the quality model assessment. This assessment consists of aggregating the gathered data into metrics, product and process factors, and strategic indicators (see Fig. 1b). The *Strategic Decision Making* package includes the *Strategic Dashboard* component responsible of the interaction with the decision-maker.

The current version (hereafter called Q-Rapids prototype) was released in December 2017. This prototype was extensively tested, validated and evaluated agains real conditions in software development projects run by the companies providing use cases to the Q-Rapids project (four different evaluation use cases).

Next, we report the status of the two packages of the Q-Rapids tool prototype.

3.1 Data Gathering and Analysis

Data Producers. The heterogeneous sources supported collect data about static code analysis (e.g., SonarQube), executed tests during development (e.g., Jenkins), code repositories (e.g., SVN, Git, GitLab), and issue tracking tools (e.g., Redmine, GitLab, JIRA, Mantis).

Data Ingestion. It consists of several Apache Kafka[2] connectors to gather data from data producers. These connectors query the API of data producers to ingest the data into Kafka. For instance, the Jira connector reads all the features from each issue (e.g., description, assignee, due date) from the JSON document got from the Jira API[3]. Apache Kafka is a Big Data technology serving as primary ingestion layer and messaging platform, and offering scalability via clusters capabilities. This has been the more challenging module from the technical point of view. The diversity of data producers has been the main challenge, not only because of the number of tools but also because of the different versions of the same tool. We also faced the fact that some tools are used differently in the four companies where the tool has been evaluated, e.g. different metadata for issues.

[2] https://kafka.apache.org/.

[3] https://developer.atlassian.com/server/jira/platform/rest-apis/.

Distributed Data Sink. This module is used for data storage, indexing and analysis purposes. Both the raw data (i.e., collected data) and the quality model assessment (i.e., aggregations) are stored a search engine, namely Elasticsearch from the Elastic stack[4]. This allows to define four types of indexes, three for the quality model assessment elements (strategic indicators, product and process factors, and metrics), and the fourth for the raw data. As Apache Kafka, the Elastic Stack offers scalability via cluster capabilities, which is required in the multinational IT company of the Q-Rapids project.

Data Analysis and Processing. It performs the quality model assessment based on the raw data gathered following a bottom-up approach. First, raw data is used to calculate the metrics, whose calculation is normalized and interpreted after assessing the collected data. Due to such assessment, their value goes from 0 to 1, being 0 the worst value and 1 the best value regarding quality. This value come from a utility function [4], which interprets the raw data value by either the preferences of experts or learned data. Once the metrics are calculated, they are aggregated into product and process factors, and then into strategic indicators. The aggregations are computed considering the weights on child elements, and then stored in the distributed data sink.

3.2 Strategic Decision Making

The *Strategic Decision Making* package includes *Strategic Dashboard* component that provides the user interface of the Q-Rapids tool. It is a web application that consumes data from the *Distributed Data Sink* module.

The main purpose of this component is to provide an easy, attractive yet informative interface to allow decision-makers accessing the different features of the tool. Figure 3 shows the landing page of Q-Rapids Dashboard.

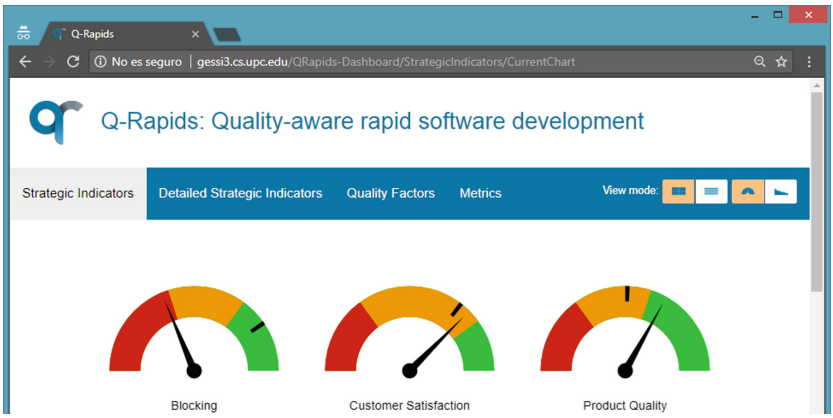


Fig. 3. Q-Rapids Dashboard landing page: Strategic Indicators View (Color figure online)

[4] https://www.elastic.co/.

Q-Rapids Dashboard includes four views, and for each view the user can choose whether seeing the current assessment or viewing the historical data; graphically or in a textual way. The four views correspond to:

- *Strategic Indicators View*: general strategic indicators status (see Fig. 3).
- *Detailed Strategic Indicators View*: for each strategic indicator, the dashboard visualizes the status of the factors affecting the strategic indicator.
- *Factors View*: for each factor, the dashboard displays the status of its metrics.
- *Metrics View*: the dashboard visualizes the metrics status.

The key feature of this tool is the aggregation of heterogeneous elements. In order to be able to aggregate different kind of data, the tool works with normalized and interpreted data (see Sect. 3.1). Therefore, the values shown by the tool are in the range 0 to 1, where 0 indicates bad quality and 1 good quality.

Figure 3 visualizes strategic indicators using gauge charts, which provides a quick visual trouble identification mechanism for decision-makers. The speedometer needle in the red zone indicates a potential risk, and in the green zone the strengths. The black mark in the gauge indicates the target value to reach. Figure 4 shows alternative ways to visualize strategic indicators. From left to right, there are graphical views to visualize all the factors impacting in a strategic indicator using radar charts (left), charts visualizing the historical data, showing the evolution of the strategic indicators (middle), and the evolution of factors impacting in it (right).

Fig. 4. Alternative views to visualize strategic indicators

In order to facilitate the analysis and the understanding of the status of the strategic indicators assessment, the user can navigate forward and backwards from the different levels of abstraction views in the following order:

Strategic Indicators ↔ Detailed Strategic ↔ Factors ↔ Metrics.

A complete description of the dashboard functionality is available as User's Guide that can be downloaded from the Q-Rapids project website (downloads section), jointly to a video tutorial[5] of the dashboard.

[5] https://youtu.be/2m-dmJZiYBA.

4 Tool Evaluation

We designed a semi-structured interview to evaluate the Q-Rapids prototype in January 2018. We aimed at understanding amongst others its *usability*, *ease of use*, and *relevance* from the perspective of *product owners* and identifying needs for improvements. We measured usability, ease of use, and relevance using the Likert-scales defined in [7, 8]. Each Likert-scale includes up to four statements to be rated using a response scale from 1: strongly disagree to 5: strongly agree.

Before each evaluation, we selected one project per industrial partner, configured and installed the Q-Rapids prototype, and collected the corresponding project data for a period of not less than 2 weeks. Then, we performed individual evaluations with eight product owners from the four companies involved in Q-Rapids project. Each evaluation session includes four steps. After explaining the study goals and procedures, we trained each participant in the Q-Rapids prototype using the video mentioned above. Then, we asked the participant to analyze the status of the project's strategic indicators, quality factors, and metrics using the Q-Rapids prototype. We encouraged the participant to think aloud and mention both positive and negative aspects of the Q-Rapids prototype. Finally, we asked the participant to answer a feedback questionnaire on the usability, ease of use, and relevance of the Q-Rapids prototype.

More than half of the participants ($n = 5$) consider the Q-Rapids prototype as moderately usable ($Mdn = 3.25$, $Mode = 3$, $Min = 2.5$, $Max = 5$). They perceive the in-formation provided by the Q-Rapids prototype as useful. However, they claim there is a need for linking the strategic indicators, quality factors and metrics with other information sources (e.g., source code, user stories, and list of issues) in order to better support the decision making process. The participants agree that integrating several data sources is an added value for supporting the decision making process in their companies. The majority of the participants ($n = 7$) considered the Q-Rapids prototype as easy to use ($Mdn = 4$, $Mode = 4$, $Min = 3$, $Max = 5$). They recommended adding functionalities for sorting values and filtering information by selecting time periods or project milestones would further increase the ease of use of the Q-Rapids prototype. Furthermore, more than half of the participants ($n = 5$) considered the Q-Rapids tool as relevant ($Mdn = 4$, $Mode = 4$, $Min = 3$, $Max = 4$). They commented the prototype has high potential to support a closer work between managers and the developers.

The evaluation results are only an indication and cannot be generalized because of a convenient sample of participants used the Q-Rapids prototype to solve few tasks in a controlled environment

5 Roadmap

There are several tools in the market for aggregating and visualizing data in a graphical way. For example, software quality tools (SonarQube, Black Duck, Bitergia), Business Intelligence tools providing dashboards (Tableau, Microsoft Power BI, Pentaho) and reports (ReportServer, JasperReports, BIRT). The common way of working of these tools is that the organization using the tool should customise their own visualizations depending on their data, Q-Rapids method and tool face this customization at level of

data, i.e. designing the quality model and the quality model is visualized through a generic dashboard. Giving us the opportunity of adding analysis capabilities over the quality model. Additionally, we envisage the Q-Rapids Tool as a more powerful tool with specific capabilities to support decision-making in managing quality in rapid software development. Next releases of the tool are planned for August 2018 and August 2019.

The new features planned for the next release are: (1) the use of Bayesian networks [9] to estimate the strategic indicators assessment, (2) what-if analysis techniques, (3) candidate QR suggestions, and (4) collection of data at run-time.

Besides the new features, we will include some improvements suggested by the industrial partners during the evaluation. One of the most demanded improvement has been the access to the raw data. We will materialize this request allowing decision-makers to drill-down until raw data, giving them the option to have a deeper analysis arriving to the source of the problem.

6 Conclusions

Q-Rapids Tool is a data-driven tool that allows decision-makers managing the quality of their products. The Q-Rapids prototype provides four sets of functionality: (1) *data gathering* from several and heterogeneous data source tools: project management (GitLab, Jira, Mantis, Redmine), software repositories (Git, GitLab, SVN), code quality (SonarQube), and continuous integration (Jenkings); (2) *calculation and aggregation of data* into three levels of abstraction (metrics, product and process factors, and strategic indicators) shaping a quality model containing preferences of experts or learned data; (3) *visualization of the aggregated data* (current and historical); and (4) *navigation through the aggregated data.* The different levels of abstraction in the quality model support decisions at different levels in organizations. The visualization functionalities include the current and historical data that can be displayed graphically or in textual form. The historical data support decision-makers to make trend analysis to anticipate risks. The dashboard includes drill-down capabilities making possible to visualize the behavior of strategic indicators allowing to visualize the reasons behind a bad assessment (i.e. which metric is affecting negatively).

The evaluation results of the first Q-Rapids prototype indicate that product owners perceive it as easy to use and relevant. However strategic indicators, quality factors, and metrics have to be linked with further information (e.g., source code and product backlog) to better support the decision making process. We plan to evaluate subsequent versions of the Q-Rapids prototype by performing case studies.

Acknowledgments. This work is a result of the Q-Rapids project, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 732253. We would like to thank Guillem Bonet, Oriol Martínez, and Axel Wickenkamp for the implementation of the Q-Rapids prototype.

References

1. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 35–46 (2000)
2. Schön, E.M., Thomaschewski, J., Escalona, M.J.: Agile requirements engineering: a systematic literature review. Comput. Stand. Interfaces **49**, 79–91 (2017)
3. Guzmán, L., Oriol, M., Rodríguez, P., Franch, X., Jedlitschka, A., Oivo, M.: How can quality awareness support rapid software development? – a research preview. In: Grünbacher, P., Perini, A. (eds.) REFSQ 2017. LNCS, vol. 10153, pp. 167–173. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54045-0_12
4. Wagner, S., et al.: Operationalised product quality models and assessment: the Quamoco approach. Inf. Softw. Technol. **62**(2015), 101–123 (2015)
5. Martínez-Fernández, S., Jedlitschka, A., Guzmán, L., Vollmer, A.M.: A quality model for actionable analytics in rapid software development (2018). arXiv:1803.09670
6. Kozik, R., Choraś, M., Puchalski, D., Renk, R.: Q-Rapids framework for advanced data analysis to improve rapid software development. J Ambient Intell. Humaniz. Comput. (2018). https://link.springer.com/article/10.1007/s12652-018-0784-5
7. Venkatesh, V., Bala, H.: Technology acceptance model 3 and a research agenda on interventions. Decis. Sci. **39**(2), 273–315 (2008)
8. McKinney, V., Yoon, K., Zahedi, F.M.: The measurement of web customer satisfaction: an expectation and disconfirmation approach. Inf. Syst. Res. **13**(3), 296–315 (2002)
9. Jensen, F.V.: An Introduction to Bayesian Networks. UCL Press, London (1996)

A NMF-Based Learning of Topics and Clusters for IT Maintenance Tickets Aided by Heuristic

Suman Roy, Vijay Varma Malladi(✉), Abhishek Gangwar, and Rajaprabu Dharmaraj

Infosys Ltd., # 44 Electronics City, Hosur Road, Bangalore 560100, India
sum_roy@yahoo.com,
{VijayVarma_Malladi,Abhishek.Gangwar,Rajaprabu.Dharmaraj}@infosys.com

Abstract. Ticketing system is designed to handle huge volumes of tickets generated by large enterprise IT infrastructure components. As huge amount of information remain tacit in the ticket data, topic derivation from them is essential to extract information automatically to gain insights for improving operational efficiency in IT governance. Because tickets are short and sparse by nature existing topic extraction methods cannot be suitably applied to them to learn reliable topics (and thus clusters). In this paper, we propose a novel way to address this problem which incorporates similarity between ticket contents as well as some heuristic. Topic derivation is done through a 2-stage matrix factorization process applied to ticket-ticket and ticket-term matrices. Further tickets are assigned to a topic (and the corresponding cluster) based on the derived dependence of tickets on topics and terms using some heuristic. We experiment our approach with industrial ticket data from several domains. The experimental results demonstrate that our method performs better than other popular modern topic learning methods. Also the use of heuristic in clustering does not affect the labeling of topics as the same set of labels are reproduced with similar scores.

Keywords: IT governance · Tickets · Summary · NMF · Topics Clusters · Silhouette index · Davies-Boudin index · Anonymous topic Labels

1 Introduction

Hundreds of tickets are raised everyday on a Ticketing system which is an input for Information Technology Infrastructure Library (ITIL) services such as problem management and configuration management. The incident tickets record symptom description of issues, as well as details on the incident resolution. If tickets are grouped into different clusters based on their textual content they can

S. Roy—The work was done when the author was with Infosys Ltd.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 209–217, 2018.
https://doi.org/10.1007/978-3-319-92901-9_18

provide a better understanding of the types of issues present in the system. This clustering of tickets can be achieved through topic modeling. Such topic learning is done by using popular topic derivation methods including Latent Dirichlet Allocation (LDA) [2], Probabilistic Latent Semantic Analysis (PLSA) [6], Non-negative Matrix Factorization (NMF) [8] etc.

Ticket summaries are short in nature which are restricted to 2–3 sentences. Hence, the frequency of co-occurring terms among tickets is normally low. This results in an extremely sparse relationship matrix between the tickets and the terms occurring in them. Like Twitter data, the normal topic derivation methods using this matrix will generate poor quality topics with traditional topic learning algorithms. To alleviate this problem we propose a novel approach to derive topics. This approach called T2-NMF, works in 2 stages. First, we build a ticket-ticket relationship matrix by considering content similarity of each pair of tickets. Further we subject this matrix to symmetric NMF (Non-negative Matrix Factorization) to produce ticket-topic matrix. By this matrix each topic can be represented as a non-negative linear combination of tickets and each ticket is represented as an additive combination of base topics. In the second stage we factorize the ticket-term relationship matrix as a product of ticket-topic matrix (using previously obtained) and topic-term matrix using NMF again. This matrix produces each topic as a non-negative combination of terms. From ticket-topic relationship matrix the cluster membership of each ticket can be easily determined by finding the base topic with which the concerned ticket has the largest projection value assuming that each topic corresponds to exactly one cluster under T2-NMF approach.

By assigning tickets to topics and thus clusters, in the above-mentioned way does not produce the desired result. A closer look at the clusters reveal that quite a few clusters contain heterogeneous tickets and correspondingly, their Silhouette indices are low. As our aim is to generate a coherent topic model of tickets in which only similar tickets would be grouped in the same cluster we adopt a heuristic-based approach on top of T2-NMF approach, which will be called hT2-NMF. By this we shall assign a ticket to a topic (and hence a cluster) based on its contribution to the topic content. If this value of contribution is no less than a threshold value then we assign the ticket to this topic, otherwise the ticket is bucketed into an anonymous (unnamed) cluster, the idea being that if a ticket cannot be assigned properly to a topic then we leave it unlabeled. This way we have been able to get more homogeneous clusters of tickets and an improved value for their Silhouette indices. Moreover, our experiment shows that in most of the cases the labels that are generated and chosen for the clusters using T2-NMF remain the same while we use hT2-NMF, and attain similar scores. The details of this method can be found in [13].

1.1 Related Work

A vast amount of literature is available on learning topics out of lengthy documents using techniques like PLSA [6], LDA [2], NMF [14] etc. These techniques

do not produce fruitful results for short texts like tweet data, as term document matrix is very sparse. As a consequence, the quality of topics suffers. Although some extensions of them have been proposed to deal with tweets [3,4,12] sparsity still remains a problem. To deal with this problem some studies have used 2-step approach to derive topics out of tweet data. For example, researchers in [3] have created term correlation matrix by computing the positive mutual information value for each pair of unique terms. The term correlation matrix is then factored along with the tweet-term matrix to learn topics for the tweets and keyword representation for topics. In another work [12] the authors use tweet text similarity and interaction measure to derive tweet-tweet relationship matrix which is then factorized along with tweet-term matrix to obtain the clusters of tweets and the topic representation of keywords. In this work we adopt similar techniques to learn topics out of ticket data. We factorize ticket-ticket relationship matrix along with the ticket-term matrix to derive the topics (and hence clusters) for tickets and the term representation of tickets. Additionally we incorporate the interaction between ticket vectors (using term frequency) and the relevant topic vectors by using suitable heuristic to finally assign tickets to topics.

2 Ticket Model

We consider incident tickets with similar schema which are frequent in IT maintenance. These tickets usually consist of two fields, fixed and free form. Fixed fields are customized and inserted in a menu-driven fashion. Example of such fields are the category of a ticket, sub-category, application name, incident type etc. There is no standard value for free-form fields. These fields can contain description or summary of a ticket which captures the issue behind the creation of such ticket on the maintenance system. This can be just a sentence that summarizes the problem reported in it, or it may contain a short text describing the incident.

The fixed field entries of a ticket can be represented using a relational schema. For them we shall consider only a limited number of fixed fields of a ticket for choosing attributes that reflect its main characteristics (the domain experts' comments play an important role in choosing the fixed fields). They can be represented as a tuple: Ticket (application name, category and sub-category). Each of the instances of tuples corresponding to entries in the fixed fields in the ticket can be thought of an instantiation of the schema.

We consider the collection of summary of tickets for extracting their features using light natural language processing. As a pre-processing we remove the tickets which do not contain a summary. We delete unnecessary characters from a summary so that it contains only alphanumeric characters. We delete stop words from the collection of summary and then use Stanford NLP tool to tokenize the summaries and perform POS tagging on the tokens. Once the POS tagging is performed we lemmatize the tokens. All the nouns are added as unigrams (also called keywords). Certain combinations of adjectives, verbs and nouns are used to form the bigrams and trigrams with the help of some heuristics such as *All Words Heuristic*, *Any Word Heuristic*.

The final list of keywords and keyphrases (together they will be called terms also) consists of unigrams, bigrams and trigrams extracted as above. We discard terms with DF (Document frequency) smaller than 3 to remove some rare terms and some noise which do not contribute to the content of ticket summary.

Thus we can model a ticket as a vector $T = (x_1, \ldots, x_m)$, where each element $x_l, 1 \leq l \leq m$ represents the importance or the weight of a term l with respect to the ticket T. Here we take $x_l = tf * idf(T, l)$ where $tf * idf(T, l)$ denotes the TF * IDF of a term l *wrt* ticket T as its weight (here we assume smoothened IDF).

3 The NMF-Based Approach

We assume that a collection of tickets can be grouped into k clusters each of which corresponds to a coherent topic. In this work we assume hard clustering in which each ticket is assigned to one cluster only. In normal practice one uses non-negative matrix factorization (NMF) on ticket-term matrix to factor into ticket-topic and topic-term matrix. While the former matrix captures each ticket as a non-negative linear combination of topics the latter matrix produces a topic as an additive mixture of terms.

However, the ticket-term matrix which captures term occurrence is extremely sparse as each ticket contains very few lines of text [3,12]. To alleviate this problem, we propose a novel approach to derive topics from a collection of tickets by considering ticket content similarity through a two stage method, for details see [13].

Relationship Between Tickets. Recall a ticket T_i can be represented as a vector $T_i = (x_{i1}, \cdots, x_{im})$, where x_{ip} indicates the weight (typically TF * IDF value) of term t_p in Ticket T_i. Ticket T_j can be represented likewise $T_j = (x_{j1}, \cdots, x_{jm})$. We define the cosine similarity between two tickets T_i and T_j as:

$$\text{sim}_C(T_i, T_j) = \frac{\sum_{p=1}^{m} x_{ip} * x_{jp}}{\sqrt{\left(\sum_{p=1}^{m} x_{ip}^2\right) + \left(\sum_{p=1}^{m} x_{jp}^2\right)}}$$

Assuming that we have n tickets in the collection we define a symmetric matrix $\mathbf{R} = [r_{ij}]$ of degree n as $r_{ij} = 1$ if $i = j$. Otherwise $r_{ij} = \text{sim}_C(T_i, T_j)$. This $\mathbf{R}$ is called *ticket-ticket relationship* matrix or simply, *ticket-ticket* matrix.

Assuming that a ticket is represented as a vector of terms as above we build the ticket-term matrix denoted as $\mathbf{X} = [x_{ij}]$ of dimension $n \times m$, where x_{ij} represents the TF*IDF value of term j for ticket i.

Topic Learning for Tickets. The ticket-ticket relationship is modeled capturing the content similarity. The derived ticket-topic matrix will model the thematic structure of the relationships between tickets. Subsequently, using techniques similar to graph clustering the topic learning problem is formulated as finding the ticket-topic matrix $\mathbf{U}$ by minimizing the following objective function: $L(\mathbf{U}) = \frac{1}{2}\left\|\mathbf{R} - \mathbf{U}\mathbf{U}^T\right\|_F^2$, s.t. $\mathbf{U} \geq 0$, where $\mathbf{U}$ is of dimension $n \times k$ (assuming a fixed

number of topics equaling to k). Further each column of $\mathbf{U}$ represents a topic by a vector of weighted contribution of each ticket. This special form of non-negative matrix factorization is known as the symmetric non-negative matrix factorization [3], which is a special case of NMF decomposition. We use a parallel multiplicative update algorithm from [5] to solve this problem. This algorithm works by first minimizing the Euclidean distance and then adopting an α-update rule.

Deriving Representative Terms for a Topic. In the second step we try to generate term representation for a topic. One obtains these representative terms by factoring the ticket-term matrix into the ticket-topic matrix and the topic-term matrix. Our method utilizes ticket-topic matrix to derive the representative terms. In particular, we consider the ticket-term matrix $\mathbf{X}$ and project the terms into the latent topic space. In the NMF framework, this problem is formulated as finding a non-negative matrix $\mathbf{V}$ (of dimension $m \times k$) by minimizing the loss function: $L_\Gamma(\mathbf{V}) = \frac{1}{2}\left\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\right\|_F^2$, s.t. $\mathbf{V} \geq 0$, and $\mathbf{U}$ is given.

The above equation is a NNLS (Non-negative Least Square) problem. Here we use multiplicative update algorithm (MUA) [7,14] for solving this NNLS (the rows for $\mathbf{X}$ are initialized to have Euclidean length of 1). However we use only one update rule as $\mathbf{U}$ is already given: $v_{ij} \leftarrow v_{ij} \frac{(\mathbf{X}^T\mathbf{U})_{ij}}{(\mathbf{V}\mathbf{U}^T\mathbf{U})_{ij}}$. For this reason we only normalize the values of entries in $\mathbf{V}$ as follows: $v_{ij} \leftarrow \frac{v_{ij}}{\sqrt{\sum_j v_{ij}^2}}$.

4 Ticket Clustering Using Derived Topics Aided by Heuristic

So far, we have learned two matrices $\mathbf{U}$ and $\mathbf{V}$ that would be useful to partition the tickets into clusters. In the ticket-topic matrix $\mathbf{U}$ each column captures a topic axis, while each row corresponds to a ticket with its elements depicting the share of each topic θ to the ticket. Let us denote the set of generated topics as Θ. Assuming each topic giving rise to a unique cluster, one can think of assigning a ticket to a cluster by examining the ticket topic matrix $\mathbf{U}$, in which each element u_{ij} denotes the fraction of the contribution of topic $\theta_j \in \Theta$ to ticket T_i. Should ticket T_i solely belong to topic θ_j then u_{ij} would be the maximum among the entries $u_{1j}, u_{2j}, \ldots, u_{kj}$. However this kind of assignment does not produce the desired results as quite a few heterogeneous tickets may be grouped in the same cluster. This might be attributed to the fact that the above u_{ij} value (for θ_j) may be small (also it is very difficult to determine a threshold value) and a ticket cannot be solely assigned to a topic (cluster) based on this value. To overcome this problem we consider the interaction of a ticket and topic through terms and adopt the following heuristic (as a mechanism to remove outlier tickets from a cluster).

In topic-term matrix $\mathbf{V}$ each element v_{ij} represents the degree to which the term $t_i \in \Gamma$ belongs to topic $\theta_j \in \Theta$. We shall utilize this information to find the contribution of a ticket T to a topic θ. For this we shall adopt a binary

representation of a ticket as $T_i^b = (y_{i1}, \cdots, y_{im})$ with m terms, where $y_{ij} = 1$ if the term t_j is present in the ticket T_i, otherwise it is zero. Subsequently, we compute the *contribution* of ticket T_i to topic θ_j as: $\xi(T_i, \theta_j) = \mathbf{y}_{i\cdot} \cdot \mathbf{V}_{\cdot j} = \sum_{k=1}^{m} y_{ik} * v_{kj}$.

For a fixed topic (and hence cluster) θ_j we omit θ_j and write $\xi(T_i)$ to denote the contribution of ticket T_i to the topic in question. We also compute the mean and standard deviation of contributions of tickets to the topic which are denoted as μ_ξ and σ_ξ respectively. Based on this, for a fixed topic θ_j we empirically fix a threshold value of the contribution as $\kappa = 0.75 * \mu_\xi + 0.5 * \frac{\mu_\xi}{\sigma_\xi}$. We arrive at such a heuristic expression by examining contributory values of different tickets assigned to different clusters (using T2-NMF method) and the mean and variances of them.

Given a tuple in the ticket schema we say a ticket T_i is *finally assigned* to the topic θ_j in the tuple if T_i is associated with topic θ_j using T2-NMF and $\xi(T_i, \theta_j) \geq \kappa$. Otherwise ticket T_i is assigned to an anonymous topic θ_{an}. We assume each tuple will have only one such anonymous topic. Finally, some of the tickets do not get assigned to any topic as they are put in θ_{an} as warranted by the heuristic.

5 Automatic Labeling of Clusters

For automatic labeling of clusters we extract semantic labels for each topic [11]. This is followed by formulating a semantic scoring function of each label *wrt* a topic corresponding to a cluster. For details *cf.* [13]. Finally we use two-layered labels for a ticket,- the first layer label corresponds to the tuple detail that the ticket belongs to (refer to Sect. 2) and for the second layer we use the generated label which would be assigned to the ticket following the method described below.

Label Generation for Topics. We shall consider meaningful phrases/terms (in terms of ngrams) as candidate labels which are semantically relevant and discriminative across topics. We extract ngrams (bigrams and trigrams) from the summary of tickets using the feature extraction technique described in Sect. 2. After ngrams have been identified we apply some suitable measure of association to check their collocation, that is, if the words in a ngram are more likely to occur together. We use a simple frequency based approach for finding collocations in a text corpus which is based on counting, the details of which can be found in [10].

Semantic Relevance Scoring. We adopt a scoring function called First-order Relevance [11] to score the generated labels against topics. We use the relevance scoring function computed using the Kullback-Leibler (KL) divergence [11], the details are in [13].

6 Experimental Framework

We conduct the performance evaluations using Infosys internal ticket data sets which follow the schema described in Sect. 2. The data sets come from five different domains: Application Maintenance and Development (AMD), Supply and

Trading (SnT), Office Supply and Products (OSP), Telecommunication (Telecom) and Banking and Finance (BnF). SnT domain has the highest number of tickets, 35014 while AMD has the lowest number of tickets, 4510. Using appropriate NLP techniques discussed in Sect. 3 we extract terms from tickets in each domain.

Our industrial data are not partitioned a-priori and hence not labeled. So we shall use a couple of metrics to determine the quality of topic derivation which do not require the data to be labeled. We consider Silhouette index (S.I.) and Davies-Boudin index (D-B.I.) [1] for this purpose. Moreover, we want to ensure that not many tickets remain unassigned to any topic for a particular tuple. For this we introduce a metric called "Mismatch Index (M.I.)". M.I. measures the ratio of number of tickets that are assigned to the anonymous topic (those are not associated with any cluster) to the total number of tickets in a tuple.

We compare our proposed T2-NMF and hT2-NMF approaches with the following approaches: NMF [8,14], pNMF [9] and LDA [2].

7 Evaluation and Discussion

We determine the number of topics a-priori by computing Silhouette indices and choosing a number for which this has the highest value [13]. For further experimentation we only select the tuples if they contain at least 50 tickets (otherwise there are not too many tickets left for analysis after performing natural language processing). First we use T2-NMF approach to derive topics from this selection. We compute S.I. and D-B.I. for each tuple by taking the average of the S.I. and D-B.I. values of the clusters in the tuple. Afterward we perform hT2-NMF on the same tuples for each of the datasets. When we apply heuristic on the clusters generated using T2-NMF we allow some of the tickets to remain unassigned and put them in a anonymous cluster (topic). We compute the fraction of tickets that are unassigned for each tuple for calculating the M.I. values and then find the mean of them over the tuples. These average M.I. values are produced in Fig. 1. For most of the cases few tickets remain unassigned as evident from average M.I. values, the maximum M.I. value being 35% witnessed for AMD and OSP datasets.

We now compare our approach with the baseline approaches on a couple of tuples from different data sets. We consider one such tuple from Telecom which has the highest number of tickets (1732) among all other tuples across all the datasets. As shown in Fig. 2 both TC-NMF and hTC-NMF perform better than the rest of the approaches on the whole. For clusters of sizes 6 and 7, hTC-NMF outperforms TC-NMF in terms of S.I. values

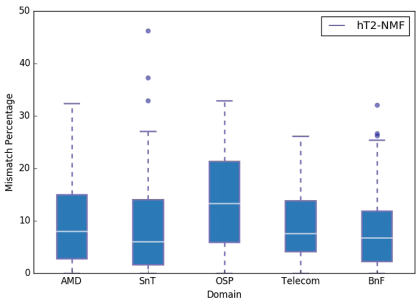
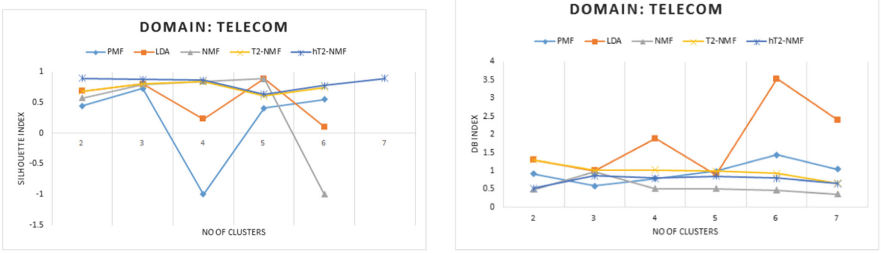


Fig. 1. Comparison of M.I. values for different domains

(Fig. 2(a)). For D-B.I. values however, for clusters of sizes 5, 6 and 7 T2-NMF performs better than hT2-NMF, but the latter performs better than the rest (Fig. 2(b)). We could find this trend in most of the tuples having higher number of tickets, but we cannot report them because of lack of space.



(a) Comparison of S.I. for a tuple in Telecom

(b) Comparison of D-B.I. for a tuple in Telecom

Fig. 2. Comparison of S.I. and D-B.I. values for a tuple from Telecom domain with different approaches

Lastly we discuss how the application of heuristic on top of NMF does not affect the labeling of clusters. We use three labels with higher scores as designated labels for a cluster (as second layer labels). We compute the mean of these scores for these three labels for each cluster in a tuple. Next we take the average of these mean scores for each tuple. This average score of each tuple is used to draw the box plot, see Fig. 3. The topmost labels almost have the same score using T2-NMF and hT2-NMF for each tuple. The median values for all domains do not change much on use of heuristic implying the divergence between the labels and the topic models remain almost unchanged.

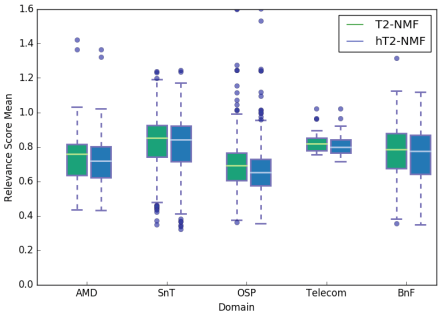


Fig. 3. Box plot for average scores of labels for tuples in all domains

8 Conclusions

Topic learning can play an important role in providing services for many applications of IT management by grouping tickets arising in their maintenance into different clusters which helps identify the root causes behind the generation of these tickets. In future we plan to propose a ticket clustering pipeline which can group the tickets in real time as they arrive in streams.

References

1. Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J.M., Perona, I.: An extensive comparative study of cluster validity indices. Pattern Recogn. **46**(1), 243–256 (2013)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Cheng, X., Guo, J., Liu, S., Wang, Y., Yan, X.: Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In: Proceedings of the 13th SIAM International Conference on Data Mining 2013, pp. 749–757 (2013)
4. Choo, J., Lee, C., Reddy, C., Park, H.: UTOPIAN: user-driven topic modeling based on interactive nonnegative matrix factorization. IEEE Trans. Vis. Comput. Graph **19**(12), 1992–2001 (2013)
5. He, Z., Xie, S., Zdunek, R., Zhou, G., Cichocki, A.: Symmetric nonnegative matrix factorization: algorithms and applications to probabilistic clustering. IEEE Trans. Neural Netw. **22**(12), 2117–2131 (2011)
6. Hofmann, T.: Probabilistic latent semantic indexing. In: SIGIR 1999, pp. 50–57. ACM (1999)
7. Kuang, D., Choo, J., Park, H.: Nonnegative matrix factorization for interactive topic modeling and document clustering. In: Celebi, M.E. (ed.) Partitional Clustering Algorithms, pp. 215–243. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09259-1_7
8. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems 13, pp. 556–562. MIT Press (2001)
9. Luo, M., Nie, F., Chang, X., Yang, Y., Hauptmann, A.G., Zheng, Q.: Probabilistic non-negative matrix factorization and its robust extensions for topic modeling. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 2308–2314 (2017)
10. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
11. Mei, Q., Shen, X., Zhai, C.: Automatic labeling of multinomial topic models. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 490–499 (2007)
12. Nugroho, R., Yang, J., Zhao, W., Paris, C., Nepal, S.: What and with whom? Identifying topics in Twitter through both interactions and text. In IEEE Trans Services Computing: A Shorter Version Appeared in 2015 IEEE International Congress on Big Data as 'Deriving Topics in Twitter by Exploiting Tweet Interactions' (2017)
13. Roy, S., Malladi, V.V., Gangwar, A., Dharmaraj, R.: A NMF-based learning of topics and clusters for IT maintenance tickets aided by heuristic. Extended version available on request through Research gate (2018)
14. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval SIGIR 2003. ACM (2003)

From Security-by-Design to the Identification of Security-Critical Deviations in Process Executions

Mattia Salnitri[1(✉)], Mahdi Alizadeh[2], Daniele Giovanella[3], Nicola Zannone[2], and Paolo Giorgini[3]

[1] Polytechnic University of Milan, Milan, Italy
mattia.salnitri@polimi.it
[2] Eindhoven University of Technology, Eindhoven, Netherlands
{m.alizadeh,n.zannone}@tue.nl
[3] University of Trento, Trento, Italy
daniele.giovanella@alumni.unitn.it, paolo.giorgini@unitn.it

Abstract. Security-by-design is an emerging paradigm that aims to deal with security concerns from the early phases of the system development. Although this paradigm can provide theoretical guarantees that the designed system complies with the defined processes and security policies, in many application domains users are allowed to deviate from them to face unpredictable situations and emergencies. Some deviations can be harmless and, in some cases, necessary to ensure business continuity, whereas other deviations might threat central aspects of the system, such as its security. In this paper, we propose a tool supported method for the identification of security-critical deviations in process executions using compliance checking analysis. We implemented the approach as part of the STS-Tool and evaluated it using a real loan management process of a Dutch financial institute.

1 Introduction

Security-by-design is a key emerging principle driving research innovation and industry development and maintenance of today's systems [2]. It includes methods, languages, techniques and tools to deal with security concerns since the initial phases of the system development and to perform verification and certification activities. In this context, business process modeling languages, such as BPMN [22], are currently used to represent the behavior of large and complex systems in terms of humans' and technical components' activities and their interactions. Extensions of such languages have demonstrated to be also effective in capturing security constraints and security policies [8,26]. Automated reasoning techniques are then used to verify whether business process models satisfy specific security and privacy policies [4,12,30], such as those imposed by national or international regulations or simply needed to guarantee a certain level of security.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 218–234, 2018.
https://doi.org/10.1007/978-3-319-92901-9_19

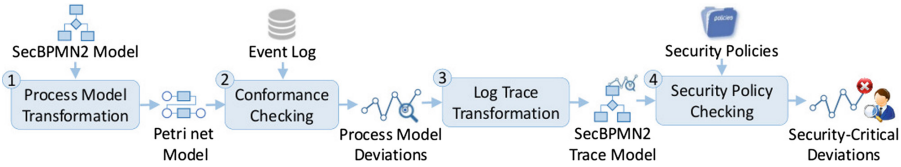


Fig. 1. An overview of the approach

Security-by-design can guarantee compliance with security policies as far as business processes are executed according to their respective (predefined) models. However, there are many application domains where deviations of the executed processes from such models is the norm. For example, in a hospital it may occur frequently that internal processes are not executed as they were designed to deal with emergency. This is perfectly natural, since when humans face unpredictable situations they require a certain level of flexibility in the execution of their activities [28,32]. It may also happen that deviations become habits for specific processes and they are never executed as they were originally designed and verified. From a security perspective, this may become a problem and it introduces the need to complement methods used to realize secure-by-design processes with methods for checking the conformance of process executions with the prescribed behavior.

Although the literature offers a number of approaches aiming to ensure that business processes comply with a set of security policies given at design time (forward compliance checking) [4,8,16,37] and techniques to evaluate the conformance of process executions with the defined process models (backward compliance checking) [1,3,6,23–25,34], what is still missing is a comprehensive framework that can close the gap between verified and certified models and their actual executions, providing an effective support to deal with security within the entire lifecycle of the system.

In this paper, we propose a tool supported method for the identification of security critical deviations in process executions, where forward and backward compliance checking are fully integrated. We adopt a state-of-art BPMN-based modeling language (i.e., SecBPMN2 [26]) to represent and verify security concerns in business processes and off-the-shelf techniques for compliance checking analysis of process executions. We show how the combination of these techniques can benefit the identification and analysis of security-critical deviations. We present an application of our approach to a real loan management process of a Dutch financial institute along with an evaluation of its capabilities. The approach is fully supported by an extended version of the STS-Tool [27].

The paper is organized as follows. Section 2 presents an overview of our approach for the identification of security-critical deviations in process executions. Section 3 demonstrates the approach in a real loan management process. Section 4 presents the tool supporting our approach and its evaluation. Finally, Sect. 5 discusses related work and Sect. 6 concludes the paper.

2 Identifying Security-Critical Deviations in Process Executions

This section describes our approach for the identification and analysis of security-critical deviations in process executions. Starting from secure-by-design processes, the approach aims to: (i) automatically analyze process executions, recorded in log traces, and identify deviations from the process specification; (ii) determine which deviations are security-critical; and (iii) visualize the identified deviations along with their security impact on the business processes. An overview of the approach is shown in Fig. 1.

The approach takes as input a set of business processes, a set of security constraints on the business processes, hereafter called *security policies*, and a set of log traces. For the definition of business processes and security policies, we rely on SecBPMN2 [26], which offers a means to define secure-by-design business processes. In particular, it provides a modeling language for the specification of business processes, called SecBPMN2-ml, that extends BPMN 2.0 [22] with security annotations, and a modeling language for the specification of security policies in the form of procedural patterns, called SecBPMN2-Q, along with capabilities to verify whether a given business process satisfies the defined policies.

An off-the-shelf backward conformance checking technique is used to identify log traces that deviate from the process specification. Deviating traces are then transformed into SecBPMN2 models to determine which deviations are security-critical. These deviations are graphically projected onto SecBPMN2 business processes for visual inspection. It is worth noting that most state-of-the-art (backward) conformance checking techniques use Petri nets for the representation of business processes. Thus, we have introduced an additional step for the automated transformation of BPMN2 process models into Petri nets. Below, we provide an overview of the main steps of our approach and underlying techniques. Then, in the next section, we illustrate these steps on a real loan management process.

Process Model Transformation. The first step of the approach aims to transform a SecBPMN2 business process into the corresponding Petri net. This transformation is required for conformance checking (second step). Petri nets [13] provide a formal representation of business processes for which several conformance checking techniques and tools have been proposed and are currently available (see, e.g., [1,24,33]). The transformation consists in generating a Petri net that reflects the control flow specified by the given SecBPMN2 business process. We base this process model transformation on the work of Dijkman et al. [14]. In particular, Dijkman's work specifies transformation rules for most BPMN 2.0 elements such as tasks and gateways. However, such rules do not support the transformation of inclusive gateways due to limitations on the synchronization of multiple control flows. To this end, we extend the transformation rules proposed in [14] along the lines suggested in [35], where inclusive gateways are transformed into (portions of) Petri nets that simulate the inclusive gateway behavior. Although this solution does not fully address the problem of synchronization, this problem is only

relevant when process models are simulated or executed. On the other hand, in conformance checking log traces are replayed on the process models and, thus, issues related to the synchronization of multiple control flows do not affect our approach.

Conformance Checking. The Petri nets generated in the previous step are used to identify process executions, recorded in log traces, that do not comply with the process specification. In this work, we adopt a backward conformance checking technique based on the notion of alignment [33]. In a nutshell, an alignment relates the events in a log trace to the activities in a run of the process, thus pinpointing the deviations causing nonconformity. If a log trace perfectly fits a Petri net, each "move" in the trace, i.e. each event recorded in the log trace, can be mimicked by a "move" in the model, i.e. an instance of a transition fired in the net. In cases where deviations occur, some moves in the trace cannot be mimicked by the net or vice versa. For instance, an activity could be executed when not allowed by the process model, resulting in a so-called *move on log*. Other times, an activity should have been executed according to the model but is not observed in the log trace. This results in a so-called *move on model*.

Log Trace Transformation. The third step consists in generating a SecBPMN2 business process for each log trace that does not comply with the process model. This step is required for the identification of security-critical deviations (the next step of the approach). The log traces recorded by IT systems often contain only information on the name of the activities that were executed along with the order of their execution. Although this information covers extremely important aspects of security (e.g., deviations from the prescribed control-flow might affect the correct enforcement of security mechanisms), log traces usually neither contain information on the security mechanisms implemented nor on the data objects used in the execution of activities. Such information, however, is necessary to verify security properties, such as the correct access to data or enforcement of security needs. To overcome this limitation, we exploit the information specified in SecBPMN2 processes. In particular, we extend log traces by including the elements that are not present in the log traces, such as gateways, data objects and security annotations, as specified in the SecBPMN2 process models. Intuitively, the activities of a log trace are mapped to the tasks in the SecBPMN2 business process by matching their labels. After that, a SecBPMN2 business process is created using the original process as a template. Once the SecBPMN2 business processes representing the deviating log traces are generated, they are annotated to highlight where the deviations took place. The identified deviations should be easy to inspect so that proper measures can be taken. The challenge lies in providing a visualization of deviations that scales well with the number of traces, and uses a notation familiar to the person who defined the original business process. We address this issue by aggregating all deviating traces into a single SecBPMN2 model and thus providing an overview of where the deviations happened. We chose SecBPMN2 for the visualization

of deviations as this minimizes the learning phase by reducing the amount of information users need to learn.

It is worth noting that our transformation of log traces in SecBPMN2 models assumes that tasks are executed as specified in SecBPMN2 models, i.e. with a correct implementation of security annotations, data access as specified in the model, etc. Such an assumption can be relaxed by considering richer logs or, possibly, different types of logs [1], which however are often unavailable or difficult to obtain. We leave an investigation of a more comprehensive extraction of security elements from logs for future work.

Security Policy Checking. Among the log traces that deviates from the process specification, we are interested in the ones that are security-critical. The identification of security-critical deviations is performed using SecBPMN2. In particular, SecBPMN2 supports the checking of business processes expressed in SecBPMN2-ml against security policies expressed in SecBPMN2-Q. In a nutshell, SecBPMN2 verifies if there exists a path in the process model that satisfies the given security policies (recall that SecBPMN2-Q policies specify constraints on the order of execution of activities). For each path, it is verified whether the security annotations in the process model are of the same type of those in the security policies and whether they are linked to the same SecBPMN2 elements. In this case, the security annotations in the security policies are verified against the security annotations in the SecBPMN2 model. If at least one security policy is not verified, then the corresponding trace is considered security-critical.

3 Approach at Work

This section describes an application of our method to the loan management process of a Dutch financial institute, taken from the 2012 BPI challenge [7]. We first introduce the case study along with the SecBPMN2 concepts that are necessary to understand the method and then we present a step-by-step application of the method. We assume the reader to be familiar with BPMN 2.0 [22] and Petri net [13] notations.

3.1 Loan Management Process

Our case study is about the analysis of an event log recording the loan management process of a Dutch financial institute, which was made available for the 2012 BPI challenge [7]. The event log contains the events recorded for three intertwined subprocesses: subprocess A specifies the handling of loan applications, subprocess O describes the handling of loan offers, and subprocess W specifies how work items are processed. For activities executed within subprocesses A and O, only events at stage complete are recorded, whereas for activities executed within subprocess W, events are recorded at stages schedule, start and complete.

Figure 2 shows the loan management process in SecBPMN2 notation. This process model is obtained based on the analysis reported in [36]. The process starts when a client submits a credit request. Then, if needed, fraud check (W_Beoordelen fraude) or first assessment (W_Afhandelen leads) are performed. Applications are then finalized by filling in additional information (W_Completeren aanvraag) and an offer is sent to the client. An offer can be canceled or created multiple times during a process execution. In different states of the process, the client might be contacted (W_Nabellen offertes and W_Nabellen incomplete dossiers) for obtaining missing information. After returning an offer, the applications might undergo further assessment (W_Valideren aanvraag). At the end, an application can be denied, approved or canceled. Note that after approving an application, the contract can still be modified (W_Wijzigen contractgegevens).

Security requirements defining the correct execution of the loan management process are captured using SecBPMN2 security annotations. These annotations are represented with an orange circle with an icon denoting the type of security annotation. The language supports eleven types of annotations. Here, we only present the four that are used in the case study and refer interested readers to [26] for a complete description of SecBPMN2 security annotations.

Many organizations and, in particular, banks and financial institutes, require that sensitive activities are performed by different users to prevent conflicts of interest [17,29]. In our case study, it is required that the handling of loan applications and offers is handled by at least two different employees. In SecBPMN2, this can be specified through a Separation of duties security annotation ( ) connecting two pools (or two lanes). This annotation indicates that the activities in the two pools (lanes) cannot be executed by the same person. In the process of Fig. 2, the security annotation specifies that Customer Service and Specialist cannot be the same person in an execution of the process.

Moreover, it should not be possible to challenge the execution of some activities, for example the approval (A_APPROVED) or denial (O_DECLINED) of a loan in Fig. 2, and the validity of the associated contracts. To this end, a legal proof of the execution of those activities is usually requested. Such a proof demonstrates, from a legal point of view, that the activity has been executed and no one can deny its execution. In SecBPMN2, this is specified with a Non repudiation security annotation ( ), which denotes that a legal proof of the execution of an activity has to be generated.

Some activities can be central to the core businesses of an organization. Such types of activities are captured using Availability security annotations ( ), which specify that an activity should always be executed. For example, in Fig. 2, activity A_SUBMITTED is associated with an availability annotation indicating that customers should always be able to request a loan. SecBPMN2, optionally, allows the specification of an availability threshold indicating, for example, an availability of 99% of the time. Business processes may use personal data, stored in data objects, which requires special protection due to its sensitive content. SecBPMN2 uses the Confidentiality security annotation ( ) to specify that the

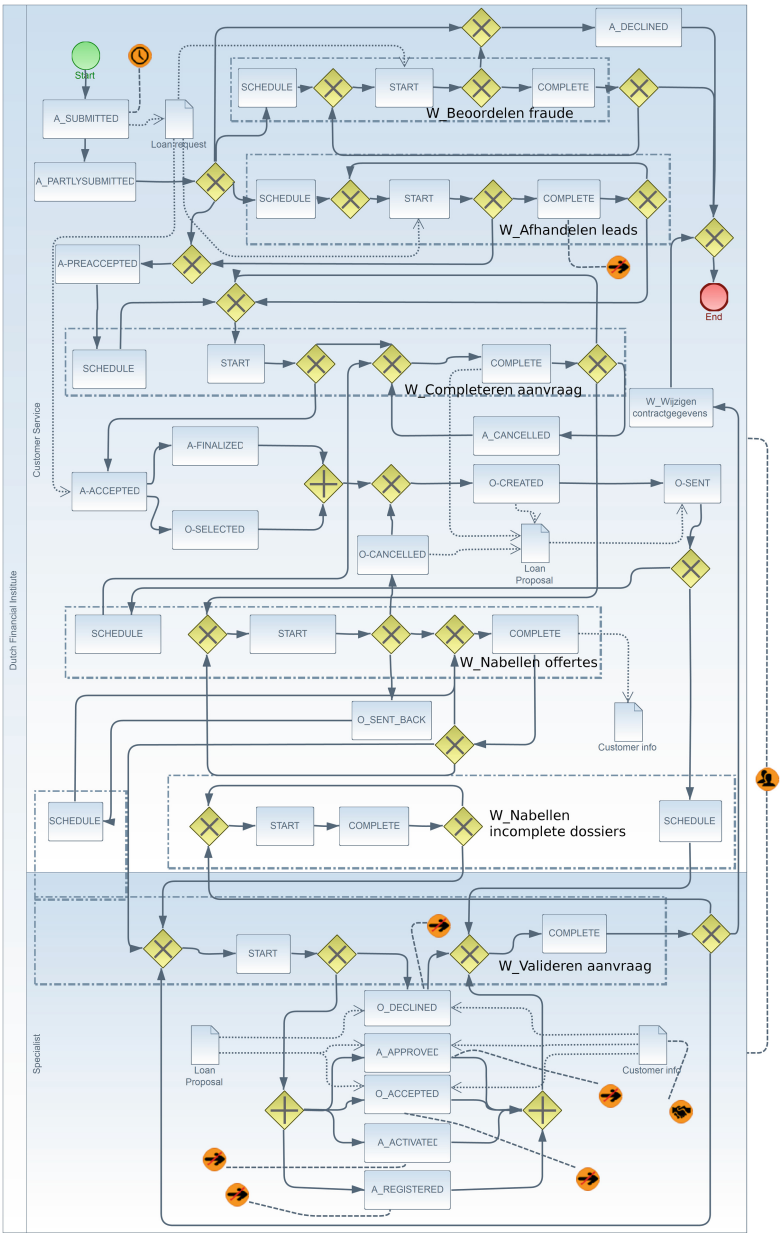
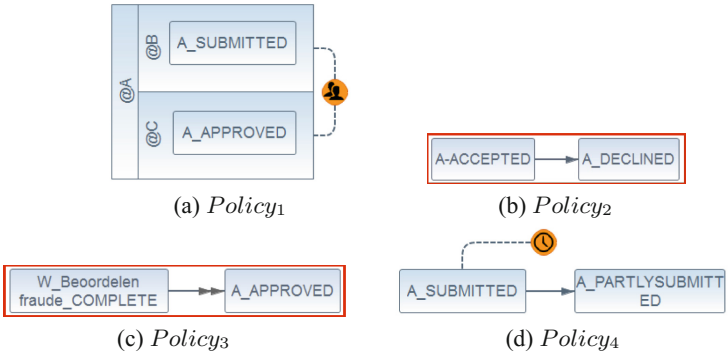


Fig. 2. Loan management process in SecBPMN2-ml notation



(a) $Policy_1$ (b) $Policy_2$ (c) $Policy_3$ (d) $Policy_4$

Fig. 3. Policies for the loan management process of Fig. 2

access to a data object must be prevented to unauthorized users. For example, in Fig. 2, data object customer info contains customer personal data and the security annotation specifies that the access to that data object should be limited only to authorized users.

SecBPMN2 also allows the specification of security policies to determine whether the defined process is secure-by-design. Figure 3 presents four sample security policies for the case study defined using SecBPMN2-Q. $Policy_1$ indicates that the handling of the submission and approval of an application must be performed by two different users. This is specified with a Separation of Duties annotation linking two lanes. The @ wild card specifies there is no constraint on the user performing the activities, i.e. the policy applies to every user. $Policy_2$ and $Policy_3$ are anti-patterns (denoted by a red box), indicating that the behavior described in these policies should not be observed in any process executions. $Policy_2$ imposes that, after an application is accepted, it cannot be immediately declined. This is specified using a sequence flow relation indicating that A_DECLINED cannot be executed immediately after A_ACCEPTED. $Policy_3$ specifies that if an application is inspected for fraud, eventually it must not be approved. This is specified using a walk relation indicating that two activities must not be linked with an arbitrary long sequence of executable elements. Finally, $Policy_4$ indicates that the handling of application submissions has to be provided with high availability (specified using the availability annotation) and its execution should be followed by the execution of A_PARTIALLYSUBMITTED. This is specified using a sequence flow indicating that the target activity must be executed immediately after the source activity.

A verification of the process in Fig. 2 against the policies in Fig. 3 shows that the process is secure-by-design, i.e. all policies are satisfied by the process model. However, reality can diverge from the prescribed behavior. Figure 4 presents three log traces recorded by the financial institute that deviate from the loan management process in Fig. 2. In the next section, we illustrate our approach for the identification of security-critical deviations in process executions. Then, in Sect. 4, we present an analysis of the log traces in Fig. 4.

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	A_PREACCEPTED-COMPLETE	112
4	W_Completeren aanvraag-SCHEDULE	112
5	W_Completeren aanvraag-START	10982
6	A_ACCEPTED-COMPLETE	10982
7	O_SELECTED-COMPLETE	10982
8	A_FINALIZED-COMPLETE	10982
9	O_CREATED-COMPLETE	10982
10	O_SENT-COMPLETE	10982
11	W_Nabellen offertes-SCHEDULE	10982
12	W_Completeren aanvraag-COMPLETE	10982
13	W_Nabellen offertes-START	10982
14	W_Nabellen offertes-COMPLETE	10982
15	W_Nabellen offertes-START	11001
16	O_SELECTED-COMPLETE	11001
17	O_CANCELLED-COMPLETE	11001
18	O_CREATED-COMPLETE	11001
19	O_SENT-COMPLETE	11001
20	W_Nabellen offertes-SCHEDULE	11001
21	W_Nabellen offertes-COMPLETE	11001
22	W_Nabellen offertes-START	11049
23	O_SENT_BACK-COMPLETE	11049
24	W_Valideren aanvraag-SCHEDULE	11049
25	W_Nabellen offertes-COMPLETE	11049
26	W_Valideren aanvraag-START	10138
27	W_Valideren aanvraag-COMPLETE	10138
28	W_Valideren aanvraag-START	10138
29	W_Valideren aanvraag-COMPLETE	10138
30	W_Valideren aanvraag-COMPLETE	10138
31	A_APPROVED-COMPLETE	112
32	A_ACTIVATED-COMPLETE	112
33	A_REGISTERED-COMPLETE	112

(a)

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	W_Afhandelen leads-SCHEDULE	112
4	W_Afhandelen leads-START	11003
5	A_PREACCEPTED-COMPLETE	11003
6	W_Completeren aanvraag-SCHEDULE	11003
7	W_Afhandelen leads-COMPLETE	11003
8	W_Completeren aanvraag-START	10929
9	W_Completeren aanvraag-COMPLETE	10929
10	W_Completeren aanvraag-START	10913
11	W_Completeren aanvraag-COMPLETE	10913
12	W_Completeren aanvraag-START	10929
13	A_ACCEPTED-COMPLETE	10929
14	A_DECLINED-COMPLETE	10929
15	W_Completeren aanvraag-COMPLETE	10929

(b)

	Activity	Resource
1	A_SUBMITTED-COMPLETE	112
2	A_PARTLYSUBMITTED-COMPLETE	112
3	A_PREACCEPTED-COMPLETE	112
4	W_Completeren aanvraag-SCHEDU...	112
5	W_Completeren aanvraag-START	
6	A_ACCEPTED-COMPLETE	11200
7	A_FINALIZED-COMPLETE	11200
8	O_SELECTED-COMPLETE	11200
9	O_CREATED-COMPLETE	11200
10	O_SENT-COMPLETE	11200
11	W_Nabellen offertes-SCHEDULE	
12	W_Completeren aanvraag-COMPLETE	
13	W_Nabellen offertes-START	11049
14	O_SENT_BACK-COMPLETE	11049
15	W_Valideren aanvraag-SCHEDULE	11049
16	W_Nabellen offertes-COMPLETE	11049
17	W_Valideren aanvraag-START	10629
18	W_Beoordelen fraude-SCHEDULE	10629
19	W_Valideren aanvraag-COMPLETE	10629
20	W_Beoordelen fraude-START	10188
21	W_Valideren aanvraag-SCHEDULE	10188
22	W_Beoordelen fraude-COMPLETE	10188
23	W_Valideren aanvraag-START	10629
24	A_REGISTERED-COMPLETE	10629
25	A_APPROVED-COMPLETE	10629
26	O_ACCEPTED-COMPLETE	10629
27	A_ACTIVATED-COMPLETE	10629
28	W_Valideren aanvraag-COMPLETE	10629

(c)

Fig. 4. Examples of log traces.

3.2 Walkthrough Application of the Approach to the Case Study

This section describes the application of the method presented in Sect. 2 to the loan management process introduced in the previous section.

Process Model Transformation. After the loan management process has been modeled in SecBPMN2-ml along with security annotations, it is automatically transformed into a Petri net. Figure 5 shows a portion of the Petri net generated from the process model expressed in SecBPMN2 notation shown in Fig. 2. Activities are transformed into transitions with an input and output places. For example, activity A_SUBMITTED in Fig. 2 is transformed into a transition with the same name along with two places, one before and one after the transition. Another example of the application of transformation rules is the exclusive gateway after activity A_PARTIALLYSUBMITTED in Fig. 2, which is transformed, in Fig. 5, in a place connected with as many transitions as the outgoing control flows. Our transformation rules provide optimizations over the transformation rules in [14] to reduce the size of the generated Petri net. For example, the transitions after the place encoding the exclusive gateway denote the activities to be executed after the gateway, instead of duplicating places and transitions as indicated in [14].

Conformance Checking. The generated Petri net is used to assess the conformity of log traces with the model. As discussed in Sect. 2, we employ an off-the-shelf alignment-based technique [33], which provides a robust approach to conformance checking able to pinpoint the deviations causing nonconformity

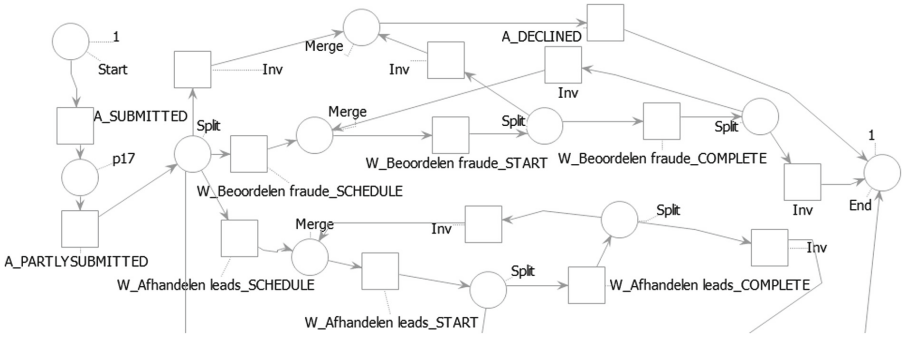


Fig. 5. Portion of the Petri net generated from the SecBPMN2 business process in Fig. 2

A_SUBMITTED	A_PARTLY SUBMITTED	W_Afhandelen leads _SCHEDULE	W_Afhandelen leads _START	...	W_Completeren aan vraag _START	≫	≫	W_Completeren aanvraag _COMPLETE	...	O_DECLINED ...
A_SUBMITTED	A_PARTLY SUBMITTED	W_Afhandelen leads _SCHEDULE	W_Afhandelen leads _START	...	W_Completeren aan vraag _START	A_ACCEPTED	A_DECLINED	W_Completeren aanvraag _COMPLETE	...	≫ ...

Fig. 6. A (portion of) alignment of the log trace in Fig. 4b and the process model in Fig. 2.

between the observed and prescribed behavior. Figure 6 shows a sample alignment between the log trace in Fig. 4b and the Petri net obtained from the SecBMPN2-ml model in Fig. 2. The top row of the alignment shows the sequence of activities in the run of the net; the bottom row shows the sequence of events in the log trace. Deviations are explicitly shown by columns that contain ≫. For example, the 7th and 8th columns in the alignment show moves on logs for activities A_ACCEPTED and A_DECLINED, indicating that these events occur in the log trace although they are not allowed according to the net. The 11th column shows a move on model for O_DECLINED, indicating that this activity must occur in the log trace according to the net but it was not executed. Other columns show that the events in the log trace match the activities in the run of the net (i.e., *synchronous moves*).

Log Trace Transformation. Log traces that deviate from the process model are transformed into SecBPMN2 models using the original SecBPMN2 business process as a template. Figure 7 shows two portions of the SecBPMN2 model generated from the trace in Fig. 4b, in which the deviations captured in the alignment of Fig. 6 are highlighted to easy inspection. In the figure, process elements are represented using the following color code: (i) the elements executed according to the process model are in orange; (ii) the control flows indicating the presence of moves on the logs are in purple; (iii) the elements for which a move on mode occurred are in brown. For example, Fig. 7a shows that some activities were executed between activities W_Completeren aanvraag_START and W_Completeren aanvraag_COMPLETE (control flows highlighted in purple). Figure 7b shows another part of the same model where activity O_DECLINED was

Fig. 7. Portion of the SecBPMN2 model corresponding to the log trace in Fig. 4b (Color figure online)

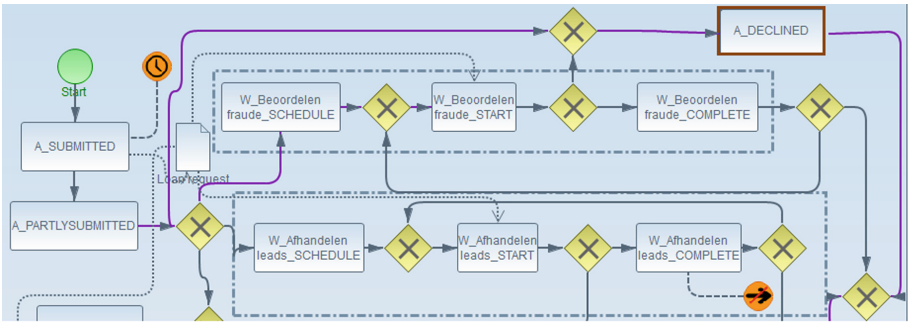


Fig. 8. Visualization of the aggregated deviating traces on the SecBPMN2 process model (Color figure online)

not executed in the log trace examined (activity highlighted in brown). Through this view, an analyst can determine where deviations occurred in single traces.

To provide an overview of the deviations in the process executions recorded in the log, we aggregate all deviating traces in a single SecBPMN2 model, thus provide analysts with an easy understanding of where deviations took place in the process. The visualization consists of the original SecBPMN2 process where the process elements for which a deviation happened in at least one process execution are highlighted in purple or brown depending on the type of deviation as shown in Fig. 8.

Security Policy Checking. The annotated SecBPMN2 model generated in the previous step shows all deviations form the prescribed behavior. In this work, we are particularly interested in security-critical deviations. To this end, deviating traces, represented as SecBPMN2 models, are verified using the SecBPMN2 verification engine to identify which log traces violate any security policy. Security-critical traces, and their aggregation, are shown to analysts using the same color code used in the previous step (see Figs. 7 and 8). An analysis of the log traces in Fig. 4 is presented in the next section.

4 Evaluation

We have implemented the proposed approach as an extension of STS-Tool [27]. STS-Tool is a software tool that permits to graphically design business processes

Table 1. Analysis of real-life data

# complaint log traces	# noncomplaint log traces	# synchronous move	# move on log	# move on model	# $policy_1$ violation	# $policy_2$ violation	# $policy_3$ violation	# $policy_4$ violation
4237	8850	232,438	29,762	28,541	3	25	30	0

and security policies using SecBPMN2, and to verify business processes against security policies. We have extended the tool by: (i) implementing the transformation of SecBPMN2 models into Petri nets; (ii) integrating alignment-based techniques to identify process executions that deviate from the process specification; (iii) adapting its verification engine to identify security-critical traces; (iv) extending the graphical editor to show aggregated traces in order to easily understand where security-critical deviations happened in business processes.

To evaluate the capabilities of our approach in the identification of security-critical deviations in real-life settings, we used the event log recording the loan management process described in Sect. 3. The event log contains 262,200 events recorded in 13,087 log traces. We applied the proposed approach to the event log, the process model in Fig. 2 and the policies in Fig. 3. Table 1 reports the results of the analysis. In total, 8,850 log traces deviated from the process model. Among the identified deviations, 29,762 are moves on log and 28,541 are moves on model. In total, 58 violations of the defined security policies were identified. To illustrate the insights provided by our the method, we next analyze the three log traces shown in Fig. 4.

Figure 4a shows a log trace that violates $Policy_1$. According to this policy, the submission and approval of a loan application have to be handled by two different users. However, in this trace, both activities were executed by the same user. By analyzing the log, we observed that this user performed many more activities than other users (17.42% of all executed activities were performed by this user) and these activities were performed every day and at any time. We speculate that this user might actually be a software application. A closer look at the log also showed that only a limited number of users (9 over 68 users) had approved a loan and the aforementioned user approved only three loan requests (as average, 249 loan applications by each user). Although we were not able to validate this finding with domain experts,[1] we believe that further investigation is required to understand the role of this user and its responsibilities and, thus, to assess the security impact of this deviation. Figure 4b shows a log trace that violates $Policy_2$. According to this policy, applications must not be declined immediately after being accepted. However, this log trace shows that after collecting all required information, the application was denied without sending any offers to the client or assessing the application. Note that the amount of the requested loan is equal to €1000. Among 361 submitted applications with equal or lower amount, only 5 applications were approved, while in general 17% of all applications were approved. This may indicate that the financial

[1] The 2012 BPI Challenge made a real log available but the log is anonymized. Also the name of the company providing the log is not disclosed.

institute tends to not grant loans with low amounts to clients and declines most of these requests. Figure 4c shows a log trace that violates $Policy_3$. According to this policy, applications suspected to be a fraud must not be approved. In fact, according to the process model, these applications must be either terminated or declined. However, this log trace shows that a suspicious application was approved.

These results confirm that our tool is able to identify security-critical deviations in real-life settings. It is worth mentioning that analyzing all deviations from the process specification is not a trivial task and can require a significant amount of time and resources. A main advantage of our tool is that it enables analysts to focus on security-critical deviations. In particular, the tool offers visualization capabilities to inspect where security-critical deviations occurred along with information about the type of security concerns (i.e., which security policy was violated). We remark that our tool aims to assist and facilitate analysts in the identification and analysis of security-critical deviations rather than taking over their responsibilities. As mentioned above, the identified deviations need to interpreted based on domain knowledge, although this task is eased by the insights provided by the tool.

The application of the tool to the case study also allowed us to perform a preliminary evaluation of its effectiveness and usability. In particular, the visualization of aggregated deviating traces into a SecBPMN2 diagram provides an intuitive view of where deviations occurred and scales well with a large number of traces, as in our case study. The visualization of single deviating traces is effective since the tool shows the business process, as drawn by the analyst, but includes only the process elements occurring in the trace. However, we realized that the usability of the tool could be improved by providing additional features for the filtering and selection of deviating traces.

5 Related Work

The goal of this paper is to guarantee the compliance of business processes during the entire lifecycle of the systems. Approaches to compliance checking in the area of business processes can be grouped into two main categories: *forward* and *backward* compliance checking. To the best of our knowledge, this is the first work that aims to reconcile these two orthogonal approaches.

Forward compliance checking aims to ensure that business processes models comply with a given set of requirements at design time. A number of approaches have been proposed to verify the compliance of process models, before they are deployed and executed. For instance, SecureBPMN [8] extends BPMN with access control and information flow constraints. It uses the hierarchic structure of the organization, in which the business process will be executed, to help security designers to define security properties such as binding of duty [17] and separation of duty [17,29]. However, SecureBPMN is limited in that it does not allow the specification of other fundamental security concepts such as confidentiality, non-repudiation and availability. Beeri et al. [4] propose BP-QL, a pattern-based

graphical query language for business processes. They also provide software tooling to determine the compliance of a business process–defined using WS-BPEL [20]–with behavioral patterns. The use of WS-BPEL, a machine-readable standard, hinders the readability of business processes, especially with real case scenarios, where business process easily reach hundreds of elements. APQL [16] is a textual query language, based on 20 predicates that can be composed to create complex queries. This approach suffers scalability issues: the definition of complex queries is challenging and error-prone due to its complexity. Moreover, to the best of our knowledge, this approach is not tool-supported. Wolter et al. [37] propose a modeling language for business processes and business security concepts, to graphically define security specifications. They also develop a framework that transforms security goals in security policies specified in XACML [21] and Rampart [31]. The framework automatically extracts specifications of security mechanisms aiming at the enforcement of security goals, but it does not allow security experts to compose security goals and, therefore, to create complex security policies. FPSPARQL [5] is a query language that allows defining queries using a formal textual language. FPSPARQL focuses on the analysis of business processes mined from action logs, hence making it impossible to directly define processes; in addition, it does not address security concerns. In this work, we have adopted SecBPMN2 for the definition of secure-by-design processes. It provides a modeling language for the specification of business processes with security annotations, and a modeling language for the specification of security policies along with capabilities to verify whether a given business process satisfies the defined policies.

Backward compliance checking aims to evaluate the compliance of process executions recorded in an event log with the prescribed behavior. Existing approaches [1,3,6,23–25,34] can be classified in two main streams, depending on the representation of the prescribed behavior. One stream comprises approaches that assess the compliance of process executions with business rules. Ramezani et al. [24] encode business rules into Petri net patterns and employ alignments to identify possible deviations from them. In recent years, many researchers have focused on conformance checking with respect to declarative models. For example, Chesani et al. [10] represent the prescribed behavior using declarative reactive business rules. This approach maps business rules to Abductive Logic Programming, and Prolog is used to check whether business rules are fulfilled. Montali et al. [19] propose to encode business rules in Linear Temporal Logic and evaluate them using automata. However, these approaches do not support time or data perspectives. Declarative constraints have been extended to support the time [18] and data [11] perspectives separately. Burattin et al. [9] propose an approach to support different process perspectives at the same time.

Another stream includes approaches that assess the compliance with respect to a process model. Among these approaches, Petkovićet al. [23] generate the transition system of a process model and verify whether a log trace corresponds to a valid trace of the transition system. Rozinat and van der Aalst [25] propose a token-based approach to measure the conformance between an event log and

a Petri net. This measurement is based on the number of missing and remaining tokens after replaying log traces on the net. Banescu et al. [3] extend the work in [25] to identify different types of deviations such as replacement and re-ordering by analyzing the configuration of missing and remaining tokens using predefined patterns. These approaches, however, do not provide accurate diagnostics on deviations. In this work, we adopt the notion of alignment [33] for conformance checking. Alignments offer a robust approach to conformance checking able to provide richer and more accurate diagnostics and have been widely used for various purposes such as performance analysis [33], process-model repairing [15] and security analysis [1].

6 Conclusion

This paper proposes a methodological approach for the identification of security-critical deviations in process executions, which leverages off-the-shelf techniques for forward and backward conformance checking. The use of forward compliance checking allows the specification of secure-by-design business processes, whereas the use of backward compliance checking allows the identification of deviations in the process executions. Moreover, we exploit the verification capabilities provided by forward compliance checking to identify those deviations that are security critical. Our approach is fully supported by an extended version of the STS-Tool. The tool offers visualization capabilities to inspect the identified deviations and provide diagnostics on security concerns (i.e., which policy was violated). This view provides analysts with insights into security issues, thus, helping them in taking the necessary measures to mitigate the impact of security incidents. We evaluated the tool with a case study using a real log recording the loan management process of a Dutch financial institute. The results of the evaluation show that our approach is able to identify a number of security concerns, and the visualization of the single and aggregated deviations help in the understanding of critical-security issues in process executions.

Although the evaluation showed that our approach is able to detect security-critical deviations in real settings, we plan to conduct a more extensive evaluation of the usability and effectiveness of the tool by involving practitioners. Moreover, we plan to extend our method in order to support the analysis of a larger range of security constraints in process executions. This includes the design of methods to extract security-related information (e.g., security mechanisms employed, accessed data objects) from log files.

Acknowledgments. This work has been partially funded by the NWO CyberSecurity programme under the PriCE project, by the DITAS project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement RIA 731945.

References

1. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Comput. Secur. **73**, 172–193 (2017)
2. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edn. Wiley, Hoboken (2008)
3. Banescu, S., Petković, M., Zannone, N.: Measuring privacy compliance using fitness metrics. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 114–119. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32885-5_8
4. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with BP-QL. Inf. Syst. **33**(6), 477–507 (2008)
5. Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H.R., Sakr, S.: A query language for analyzing business processes execution. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 281–297. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23059-2_22
6. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst. Appl. **41**(11), 5340–5352 (2014)
7. BPI Challenge 2012: Event log of a loan application process (2012). https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
8. Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: SecureBPMN: modeling and enforcing access control requirements in business processes. In: SACMAT, pp. 123–126. ACM (2012)
9. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. **65**, 194–211 (2016)
10. Chesani, F., Mello, P., Montali, M., Riguzzi, F., Sebastianis, M., Storari, S.: Checking compliance of execution traces to business rules. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 134–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00328-8_13
11. De Masellis, R., Maggi, F.M., Montali, M.: Monitoring data-aware business constraints with finite state automata. In: ICSSP, pp. 134–143. ACM (2014)
12. Delfmann, P., Dietrich, H.-A., Havel, J.-M., Steinhorst, M.: A language-independent model query tool. In: Tremblay, M.C., VanderMeer, D., Rothenberger, M., Gupta, A., Yoon, V. (eds.) DESRIST 2014. LNCS, vol. 8463, pp. 453–457. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06701-8_44
13. Desel, J., Reisig, W.: Place/transition Petri nets. In: Reisig, W., Rozenberg, G. (eds.) ACPN 1996. LNCS, vol. 1491, pp. 122–173. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-65306-6_15
14. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
15. Fahland, D., van der Aalst, W.M.P.: Model repair - aligning process models to reality. Inf. Syst. **47**, 220–243 (2014)
16. ter Hofstede, A.H.M., Ouyang, C., La Rosa, M., Song, L., Wang, J., Polyvyanyy, A.: APQL: a process-model query language. In: Song, M., Wynn, M.T., Liu, J. (eds.) AP-BPM 2013. LNBIP, vol. 159, pp. 23–38. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02922-1_2
17. Li, N., Tripunitara, M.V., Bizri, Z.: On mutually exclusive roles and separation-of-duty. ACM Trans. Inf. Syst. Secur. **10**(2), 5 (2007)

18. Maggi, F.M., Westergaard, M.: Using timed automata for a priori warnings and planning for timed declarative process models. IJCIS **23**(01), 1440003 (2014)
19. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. TWEB **4**(1), 3 (2010)
20. OASIS: Web Services Business Process Execution Language, April 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
21. OASIS: eXtensible Access Control Markup Language (XACML)Version 3.0, January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
22. OMG: BPMN 2.0, January 2011
23. Petković, M., Prandi, D., Zannone, N.: Purpose control: did you process the data for the intended purpose? In: Jonker, W., Petković, M. (eds.) SDM 2011. LNCS, vol. 6933, pp. 145–168. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23556-6_10
24. Taghiabadi, E.R., Gromov, V., Fahland, D., van der Aalst, W.M.P.: Compliance checking of data-aware and resource-aware compliance requirements. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) OTM 2014. LNCS, vol. 8841, pp. 237–257. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45563-0_14
25. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
26. Salnitri, M., Paja, E., Giorgini, P.: Maintaining secure business processes in light of socio-technical systems' evolution. In: RE Conference Workshops, pp. 155–164. IEEE (2016)
27. Salnitri, M., Paja, E., Poggianella, M., Giorgini, P.: STS-Tool 3.0: maintaining security in socio-technical systems. In: Proceedings of the CAiSE Forum, pp. 205–212 (2015)
28. Sarker, S., Sarker, S., Sidorova, A.: Understanding business process change failure: an actor-network perspective. J. Manag. Inf. Syst. **23**(1), 51–86 (2006)
29. Simon, R., Zurko, M.: Separation of duty in role-based environments. In: Proceedings of the Computer Security Foundations Workshop, pp. 183–194 (1997)
30. Störrle, H.: VMQL: a visual language for ad-hoc model querying. J. Vis. Lang. Comput. **22**, 3–29 (2011)
31. The Apache Software Foundation: Apache Rampart website. http://axis.apache.org/axis2/java/rampart/. Accessed April 2016
32. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**, 37 p. (2013). https://doi.org/10.1155/2013/507984. Article ID 507984
33. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Int. Rev. Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
34. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: an approach based on temporal logic. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005). https://doi.org/10.1007/11575771_11
35. van der Aalst, W.M.P., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(1), 5–51 (2003)
36. van der Aalst, W.M.P., Verbeek, H.: Process discovery and conformance checking using passages. Fundamenta Informaticae **131**(1), 103–138 (2014)
37. Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. J. Syst. Architect. **55**(4), 211–223 (2009)

Workflow Support in Wearable Production Information Systems

Stefan Schönig[1(✉)], Ana Paula Aires[2], Andreas Ermer[2], and Stefan Jablonski[1]

[1] Institute for Computer Science, University of Bayreuth, Bayreuth, Germany
{stefan.schoenig,stefan.jablonski}@uni-bayreuth.de
[2] Maxsyma GmbH & Co. KG, Floß, Germany
{aaires,aermer}@maxsyma.de

Abstract. The Internet of Things (IoT) is the inter-networking of physical objects like electronic hardware or humans using wearable digital devices. IoT allows things to be controlled remotely across existing network infrastructures. A business process is a collection of related events, activities, and decisions that involves a number of resources. To support processes at an operational level, a Business Process Management system (BPMS) can be used. During process execution, a variety of information is required to make meaningful decisions. With the emergence of IoT, data is generated from physical devices sensing their environment that reflects certain aspects of operative processes. We introduce a toolset for an IoT-aware business process execution system that exploits IoT for BPM by providing IoT data in a process-aware way, considering IoT data for interaction in a pre-defined process model, and providing wearable user interfaces with context specific IoT data provision. The toolset has been evaluated extensively in production industry.

Keywords: Process execution · Internet of Things
Wearable interface

1 Introduction and Background

Modern production information systems are generally computerized and are designed to collect and present the data which operators need in order to plan operations within the production [1]. The connection between production and information technology leads to an ubiquity of digitally supported information processing systems. This way, all participating production objects are able to communicate based on a digital, potentially mobile infrastructure. Such a production system represents a typical *Internet of Things (IoT) system* where electronic hardware, e.g., sensors and actuators are tightly integrated with human operators. One way of integrating human users into a cyber-physical system seamlessly is to equip them with *Wearable Computing Systems (Wearables)* like smartwatches [2]. These mobile devices support users in their operative tasks by directly proving them with most recent information about the production process.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 235–243, 2018.
https://doi.org/10.1007/978-3-319-92901-9_20

A business process is a collection of activities and decisions that involves a number of (human) resources. To support processes at an operational level, a BPM system (BPMS) can be used. A BPMS deals, a.o., with the enactment of models that define the interplay between environmental circumstances and activities to be executed. With the emergence of IoT, data is generated from physical devices sensing their manufacturing environment that reflects certain aspects of processes. Accordingly, sensing and perception the process environment via sensors constitutes the fundamental task of the IoT. Sensor data then must be aggregated, interpreted and made available to the BPMS in order to trigger activities. The resulting user tasks of the triggered activities must then be send in real time to responsible individuals that receive tasks on mobile user interfaces.

Consider a production process where material is processed by machines under the supervision of human operators. In case of product quality issues, human involvements are necessary. Operators must be aware of sensor data to decide on tasks to be executed next. Such a scenario might be better manageable when linking digital production data with human operators as enabled by the integration of IoT and BPM. The necessity of human activities can be triggered by a BPMS through the reference of IoT sensor data in the underlying process model [3]. This way, operators can be notified seamlessly without any loss of time on wearable devices while leveraging current context specific information. As a consequence, the integration of IoT and BPM technology could lead to efficiency gains by reducing reaction time and enhance the quality of task execution. For this purpose, we introduce a toolset for an IoT-aware process execution system that exploits IoT for BPM by providing IoT data in a process-aware way, considering IoT data for interaction in a pre-defined process model, and providing wearable user interfaces with context specific IoT data provision. The prototypical implementation has been evaluated extensively in production industry.

This paper is structured as follows: Sect. 2 describes the requirements and preliminaries for an IoT-aware process execution. In Sect. 3 we describe the prototypical implementation of our approach based on well-known communcation protocols. In Sect. 4 we describe the application of the tool in production industry. Section 5 gives an overview of related work and Sect. 6 concludes the paper and gives an outline on the tool demonstration.

2 Requirements for an IoT-Aware BPMS

In this section, we describe fundamental requirements preliminaries and that we considered in our tool. The first step is connecting IoT objects and their latest values to a BPMS. In the second step, process models need to be extended with data variables participating in the process stemming from IoT objects such as machine status or actor positions. The resulting process models must be applicable by default contemporary execution engines. This way, organizations can reuse existing process models, without having to learn new languages and remodel processes from scratch. The third step is to establish a real time notification interface

of triggered activities to process participants by means of mobile devices. In the fourth step, context relevant information stemming from connected objects must be selected and provisioned to users.

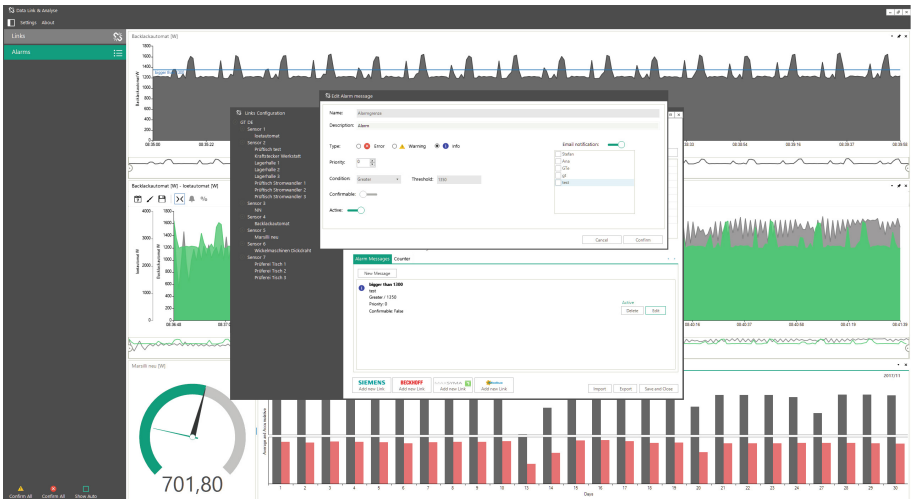


Fig. 1. IoT data acquisition, visualisation and analysis

We assume that a process model is defined as BPMN conform process diagram, one of the most used formalisms for process modeling, representing the IoT aware process to be executed. A BPMN process diagram specifies which activities are executed in a process as well as their control flow relationships. However, to be able to infer when activities start or end based on the state of IoT sensor values, the diagram must capture this information. Therefore, this step requires the process designer to enrich the given BPMN diagram by including information on how and where the connected IoT sensor data influences the process.

It is important that operators are seamlessly notified when human interaction is required, independent of where the user is located. This requires a real time notification on mobile devices of responsible users. Therefore, it is necessary to define a mapping of users to corresponding mobile devices that serve as wearable, user-specific process cockpits and task lists, respectively. This is achieved by specifying a dedicated mobile device identifier for each defined user in the BPMS. During process execution, the currently available tasks for a specific process participant are then directly sent to the specified mobile device. The actual operator to device mapping and the task distribution is described in detail in Sect. 3. Alongside with activities, context-specific and process relevant information must be provisioned to operators to improve the quality of task execution.

3 Toolset Architecture and Implementation

The IoT-aware BPMS is implemented as a three layer architecture that is visualised in Fig. 3. It consists of the following layers: *(i)* IoT objects, i.e., sensors as well as wearable devices; *(ii)* IoT infrastructure and communication middleware; and *(iii)* the BPMS. The layers are connected based on standard communication protocols. To allow IoT objects at layer *(i)* to communicate with the IoT middleware at layer *(ii)* and the BPMS, a Message Queue Telemetry Transport (MQTT)[1] Broker is used. MQTT is a publish/subscribe protocol, which is especially suited for applications where computing power and bandwidth are constrained. The used MQTT topics are listed in Table 1. IoT objects, i.e., sensors or actuators, represent publishers. They are connected to an IoT gateway using specific architectures such as Profibus, LAN, WLAN or Bluetooth. A specific IoT variable v_x is acquired and published on a MQTT topic $/v_x/data$.

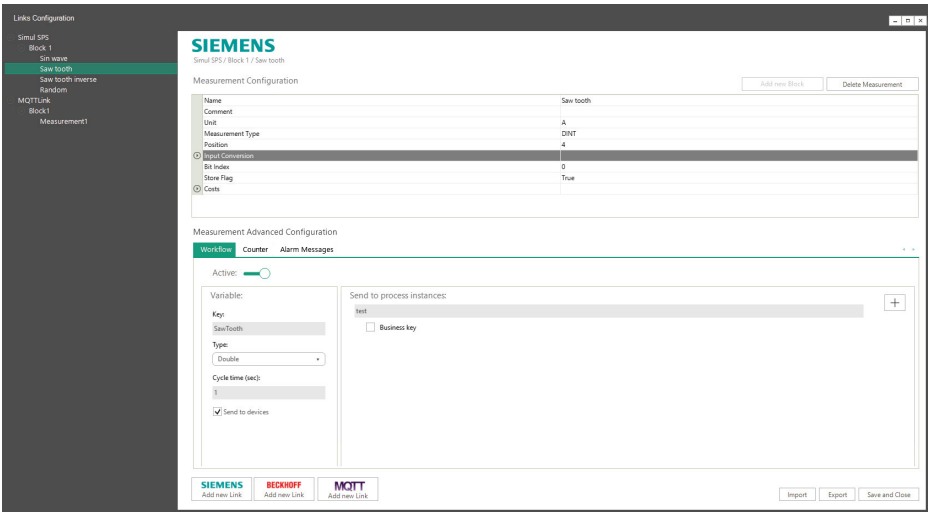


Fig. 2. IoT data distribution to BPMS and specific processes

Through a MQTT Broker the acquired data is sent to an acquisition application at layer *(ii)* that stores IoT data into a high performant NoSQL database. We used the latest version of the Apache Cassandra database. Figure 1 shows the interface of the C# based IoT data analysis desktop application where different IoT sensors can be configured. The application supports the acquisition of data stemming from PLC devices as well as from devices that communicate via MQTT. The acquired data can be visualised, accumulated and analysed. Within this application, a distribution module keeps the BPMS updated with the latest IoT values. The configuration interface is shown in Fig. 2. Here,

[1] OASIS, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

the user can select the process definitions where a certain IoT sensor variable should be send to. All running instances of the selected processes receive the corresponding data value. The application cyclically acquires the values from the database in a Key-value structure and sends them to the BPMS. In our architecture we used the latest version of the Camunda BPMS and therefore communicated with the workflow engine by means of the Camunda Rest API[2], i.e., *PUT, POST* and *GET* HTTP requests as described in Fig. 3. The tools at layer *(ii)* ensure that process relevant information stemming from the IoT is up-to-date. Through the acquistion tool, IoT data meta information is provided that makes clear where the data stems from. Given the current IoT data values, the engine calculates available activities. As a mobile user interface we implemented an *Android* based smartwatch application that subscribes to specific MQTT topics. The distribution application cyclically requests the current user tasks from the Camunda API for each defined user and publishes to the correct MQTT topic, given the mobile device identifier, i.e., smartwatch device, configured on the BPMS. The process of the device recognizing its configuration is implemented as follows: the distribution application cyclically checks the user configuration in the BPMS. When a change is detected, it publishes the new configuration to the topic /{*actor_id*}. The smartwatch of a certain actor subscribes to the topic of its specific device identifier. Having established such connections, the smartwatch communicates with the MQTT broker by subscribing to the following topics: the current tasks for a specific operator are published on the topic /{*actor_id*}/*tasks*. The device sends operators commands, such as *complete task* to the topic /{*actor_id*}/*command*. The content of the message is forwarded straight to the BPMS using a *POST* request. Context-specific IoT data is sent to actors on topic /{*actor_id*}/{*variable_id*}. To prevent the MQTT service at the watch to be killed, we implemented a keep alive communication (topic = */keepalive*).

Table 1. MQTT communication between IoT objects and BPMS

Topic	*Description*	*Direction*
/{*variable_id*}/*data*	IoT sensor data	IoT to BPMS
/{*actor_id*}	Device configuration	BPMS to IoT
/{*actor_id*}/*tasks*	Tasks of specific actor	BPMS to IoT
/{*actor_id*}/{*variable_id*}	Context data for specific user	BPMS to IoT
/{*actor_id*}/*command*	Actors actions (claim, complete, cancel)	IoT to BPMS
/keepalive		IoT to BPMS

[2] Camunda Rest API, https://docs.camunda.org/manual/7.8/reference/rest/.

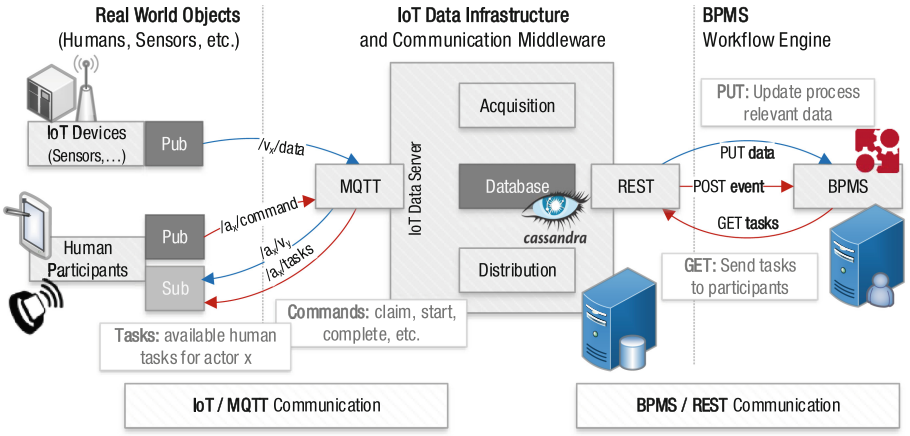


Fig. 3. Integrated communication architecture for IoT and BPMS

4 Industrial Use Cases and Prototypical Applications

We applied and evaluated the approach and toolset by means of two exemplary applications in production industry. At first we applied the prototype in the field of corrugation industry, where paper is glued to corrugated paper that depicts the basis for cardboard boxes. Due to increasing automation and staff reduction, less operators are available to control a production line. Interactions between users and machinery requires several location changes of users between control panels that result in delayed information flows. These delayed reaction times are frequently the reason for increased deficient products. To directly notify operators when human actions are needed, plant personal has been equipped with smartwatches as shown in Fig. 4. Furthermore, a user-group model has been defined in the BPMS. Here, available operators were assigned to a specific area of production that depicts their area of responsibility. Thus, depending on the area operators are working, the BPMS assigns a different set of tasks. Operators are then pointed to new human tasks through visual, acustic and, in case of noisy environments through haptic signals like vibration alarms. Furthermore, operators are used more effective because low priority work is aborted in order to perform high priority work that could lead to machine stops. In addition to currently available human tasks the IoT infrastructure provides diverse context-specific information on the smartwatch interface of operators. Depending on the specific group a user is assigned to, the wearable device offers diverse context information to operators: at the *Dry-End* (the area where produced corrugated paper leaves the plant), e.g., the remaining time of current production job, the remaining time to next stack transport, or the current production speed. Users at the *Wet-End* (the area where original paper is inducted to the plant) receive continuously information w.r.t. the next necessary roll change or occurring error and defects of machinery modules. In addition, operators receive error messages

and environmental information from the different plant modules. This way, concrete and goal-oriented information in error cases or warning messages for supply shortfalls can be transmitted to operators and enhance the over all process transparency and thus the quality of task execution. Through the described implementation it was possible to significantly reduce reaction time intervals. The amount of deficient products was decreased and the overall quality of the produced corrugated paper has been improved. The overall equipment downtime was significantly decreased, since problems have been prohibited or recognized in advance and were solved proactively. Hence, the overall equipment efficiency could be increased effectively.

Fig. 4. Wearables: (a) unclaim/complete task; (b) tasks; (c) info; (d) processes

In a second scenario, we modelled and executed a typical warehouse process that has been enhanced with IoT technology. Here, a transportation robot is called to a certain position in the warehouse by starting a specific process from the smartwatch as shown in Fig. 4d. Having started the process, a BPMN *ServiceTask* is called that sends a HTTP request to the wireless connected robot. The requests header contains the position of the watch that initiated the process.

5 Related Work

Several approaches have been proposed to relate IoT objects with business processes. [4] presents the Internet-of-Things-Aware Process Modeling Method (*IAPMM*) that covers requirements analysis. It extends the BPMN meta-model to model IoT-aware processes. The approach in [5] (*BPMN4CPS*) also describes an extension of BPMN in which the process logic is split into the cyber part, the controller and the physical part. Furthermore the authors extended BPMN by new task types. Some more notation concepts in BPMN for IoT are described in [6,7]. The main focus is on the modeling of real world properties. None of the described approaches provides details on how to execute these models. In [8] an

approach for implementing an IoT-aware execution system given in WS-Business Process Execution Language (WS-BPEL) is introduced. It extends BPEL by *context variables* which are automatically updated. The authors implemented a prototype which is compliant with every WS-BPEL engine. Other approaches implementing BPEL extensions are presented in [9]. The variables are updated using the publish/subscribe paradigm. Another extension for WS-BPEL (*Context4BPEL*) with features to manage context events to allow the asynchronous reception of events, query context data and evaluate transition conditions based on context data is described in [10]. In [11] the authors integrate distributed resources into WS-BPEL by formalizing a fragment of WS-BPEL together with the WSRF (Web Services Resource Framework). In [12] the authors propose an approach for enabling IoT-based agile business processes. They provided concepts for extending models by triggers for variance. The approaches in [13,14] rely on the information coming from artifacts involved in the process to understand how a process evolves. By adopting an extension of Guard-Stage-Milestone (GSM), it is possible to monitor the process even when the control flow is not respected. The work presented in [15] introduces a process engine for enabling mobile applications. However, this work does not comprise IoT related aspects.

6 Conclusion and Tool Demonstration Outline

In this paper, we described a toolset for IoT-aware process execution. The prototype has been implemented as a three layer architecture, i.e., IoT objects, data acquisition middleware and BPMS. The layers are connected based on the protocols MQTT and HTTP. The process cockpit has been implemented by means of wearable user interfaces with configurable context specific IoT data provision. The tool demonstration will show the acquistion of life data from IoT sensors and the distribution to several running processes on a Camunda BPMS. Based on IoT based conditional events, human tasks will be triggered and sent to smartwatches of users that can subsequently claim and execute activities.

References

1. Tao, F., et al.: Advanced manufacturing systems: socialization characteristics and trends. J. Intell. Manuf. **28**, 1–16 (2015)
2. Schönig, S., Jablonski, S., Ermer, A., Aires, A.P.: Digital connected production: wearable manufacturing information systems. In: Debruyne, C., Panetto, H., Weichhart, G., Bollen, P., Ciuciu, I., Vidal, M.-E., Meersman, R. (eds.) OTM 2017. LNCS, vol. 10697, pp. 56–65. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73805-5_6
3. Mandal, S., Hewelt, M., Weske, M.: A framework for integrating real-world events and business processes in an IoT environment. In: Panetto, H., et al. (eds.) On the Move to Meaningful Internet Systems OTM 2017 Conferences. LNCS, vol. 10573, pp. 194–212. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_13

4. Petrasch, R., Hentschke, R.: Process modeling for industry 4.0 applications - towards an industry 4.0 process modeling language and method. In: International Joint Conference on Computer Science and Software Engineering (2016)
5. Graja, I., et al.: BPMN4CPS: A BPMN extension for modeling cyber-physical systems. In: WETICE, pp. 152–157. IEEE (2016)
6. Meyer, S., Ruppen, A., Magerkurth, C.: Internet of Things-aware process modeling: integrating IoT devices as business process resources. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 84–98. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_6
7. Meyer, S., Ruppen, A., Hilty, L.: The things of the Internet of Things in BPMN. In: Persson, A., Stirna, J. (eds.) CAiSE 2015. LNBIP, vol. 215, pp. 285–297. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19243-7_27
8. Domingos, D., Martins, F., Cândido, C., Martinho, R.: Internet of Things aware WS-BPEL business processes context variables and expected exceptions. J. Univers. Comput. Sci. **20**(8), 1109–1129 (2014)
9. George, A.A., Ward, P.A.: An architecture for providing context in WS-BPEL processes. In: Conference of Advanced Studies on Collaborative Research (2008)
10. Wieland, M., Kopp, O., Nicklas, D., Leymann, F.: Towards context-aware workflows. In: CAiSE Workshops and Doctoral Consortium, vol. 2, p. 25 (2007)
11. Mateo, J.A., Valero, V., Dıaz, G.: BPEL-RF: a formal framework for bpel orchestrations integrating distributed resources, arXiv:1203.1760 (2012)
12. Schmidt, B., Schief, M.: Towards agile business processes based on the Internet of Things. In: Dangelmaier, W., Blecken, A., Delius, R., Klöpfer, S. (eds.) IHNS 2010. LNBIP, vol. 46, pp. 257–262. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12494-5_23
13. Meroni, G., Di Ciccio, C., Mendling, J.: An artifact-driven approach to monitor business processes through real-world objects. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 297–313. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_21
14. Meroni, G., et al.: Multi-party business process compliance monitoring through iot-enabled artifacts. Inf. Syst. **73**, 61–78 (2018)
15. Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A lightweight process engine for enabling advanced mobile applications. In: Debruyne, C., et al. (eds.) On the Move to Meaningful Internet Systems: OTM 2016 Conferences, OTM 2016. LNCS, vol. 10033, pp. 552–569. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48472-3_33

Predictive Process Monitoring in Apromore

Ilya Verenich[1,2(✉)], Stanislav Mõškovski[2], Simon Raboczi[3], Marlon Dumas[2], Marcello La Rosa[3], and Fabrizio Maria Maggi[2]

[1] Queensland University of Technology, Brisbane, Australia
ilya.verenich@qut.edu.au
[2] University of Tartu, Tartu, Estonia
{stanislav.myshkovski,marlon.dumas,f.m.maggi}@ut.ee
[3] University of Melbourne, Melbourne, Australia
{simon.raboczi,marcello.larosa}@unimelb.edu.au

Abstract. This paper discusses the integration of Nirdizati, a tool for predictive process monitoring, into the Web-based process analytics platform Apromore. Through this integration, Apromore's users can use event logs stored in the Apromore repository to train a range of predictive models, and later use the trained models to predict various performance indicators of running process cases from a live event stream. For example, one can predict the remaining time or the next events until case completion, the case outcome, or the violation of compliance rules or internal policies. The predictions can be presented graphically via a dashboard that offers multiple visualization options, including a range of summary statistics about ongoing and past process cases. They can also be exported into a text file for periodic reporting or to be visualized in third-parties business intelligence tools. Based on these predictions, operations managers may identify potential issues early on, and take remedial actions in a timely fashion, e.g. reallocating resources from one case onto another to avoid that the case runs overtime.

Keywords: Process mining · Predictive monitoring
Business process · Machine learning

1 Introduction

Predictive Process Monitoring is an emerging paradigm based on the continuous generation of predictions about the future values of user-specified performance indicators of a currently running process execution [6]. In this paradigm, a user defines the type of predictions they are interested in and provides a set of historical execution traces. Based on the analysis of these traces, the idea of predictive monitoring is to continuously provide the user with predictions and estimated values of the performance indicators. Such predictions generally depend both on: (i) the sequence of activities executed in a given case; and (ii) the values of data attributes after each activity execution in the case.

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 244–253, 2018.
https://doi.org/10.1007/978-3-319-92901-9_21

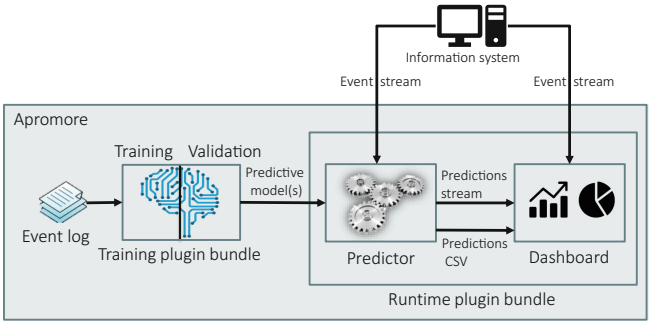


Fig. 1. High-level architecture of the predictive monitoring functionality of Apromore.

There are many scenarios where it is useful to have reliable predictions. For example, in a purchase-to-pay business process, a customer can be advised about the estimated supply date. In case this date violates service-level agreement (SLA), preemptive actions can be taken by the supplier to prevent or mitigate the delay.

A range of approaches have been proposed in the literature to tackle common predictive process monitoring tasks. A recent survey of this field [4] identified 39 distinct proposals (excluding subsumed ones) targeting time-related predictions, future path predictions and case outcome predictions. However, these approaches have largely remained in the academic domain and have not been widely applied in real-time scenarios where users require a continuous predictive support.

To fill this gap, last year we started the *Nirdizati* open-source project, aimed at developing a Web-based tool for the predictive monitoring of business processes [2]. This paper documents a major revision of the project. Following state of the art in predictive processing monitoring and numerous feedback we received from prospective users, we have redesigned the user interface, technology stack and predictive techniques behind the tool. Furthermore, both training and runtime functionality have been integrated into the process analytics platform Apromore.[1]

The two core components of Nirdizati, namely *Training* and *Runtime*, have been integrated as two bundles (i.e. sets) of plugins into Apromore (Fig. 1). The Training plugin bundle takes as input a business process event log stored in the Apromore repository, and produces one or more predictive models, which can then be deployed to the runtime predictive monitoring environment. Once a model is deployed, the Runtime plugin bundle listens to a stream of events coming from an information system supporting the process, or produced by replaying an event log stored in the repository, and creates a stream of predictions. These predictions can then be visualized in a Web dashboard or exported into a text file to be used within third-party business intelligence tools.

[1] http://apromore.org.

2 Apromore Platform

Apromore is a Web-based advanced process analytics platform, developed by the business process management (BPM) community under an open-source initiative. Apromore was originally conceived as an advanced process model repository. However, today it offers a wide range of features which go beyond those for managing large process model collections, and include a variety of state-of-the-art process mining techniques. These are techniques for the automated discovery of BPMN models, for the conformance checking of BPMN models against event logs, the replaying of event logs on top of BPMN models, the detection and characterization of process drifts from event logs, the visual analysis of process performance, and many others.

All these features are exposed through a Web portal, and organized according to the phases of the BPM lifecycle: discovery, analysis, redesign, implementation and monitoring [1]. These features can also be accessed as external Web services by third-party BPM software environments, such as ProM (for process mining) and WoPeD (for process modeling and verification).

From a technology viewpoint, Apromore relies on four core technologies: Spring, ZK, OSGi and Eclipse Virgo. Spring provides a simplified management of Java-based enterprise applications through the use of Java annotations and XML configurations. ZK is an AJAX framework used for Apromore's main Web interface (the *Portal*). OSGi provides a flexible framework for managing component dependencies through plugin bundles. Finally, Eclipse Virgo is a Web server based on the OSGi component model.

To equip Apromore with predictive process monitoring capabilities, we have wrapped the two core components of Nirdizati into two OSGi plugin bundles for Apromore: Training and Runtime. Each bundle is a set of OSGi plugins which encapsulate the logic or the user interface (UI) of the various functions offered by Nirdizati. For example, the runtime predictor is a logic plugin, while the runtime dashboard is a portal plugin (UI). These two bundles are accessible from the Monitoring menu of the Apromore Portal (see Fig. 2). One can select an event log stored in the repository, and use it to train, tune and test a variety of predictive models, by launching the training plugin bundle. Next, the runtime bundle can be used to stream an event log from the repository, or hook into a live external stream, to generate predictions as process cases unfold.

In the next sections, we introduce a working example and use this to describe the functionality of the training and runtime plugins in detail.

3 Running Example

As a running example, throughout this paper we will consider a purchase-to-pay process of an IT vendor. The process starts with lodging a purchase order and ends when requested goods have been supplied. The stakeholders are interested in predicting four variables:

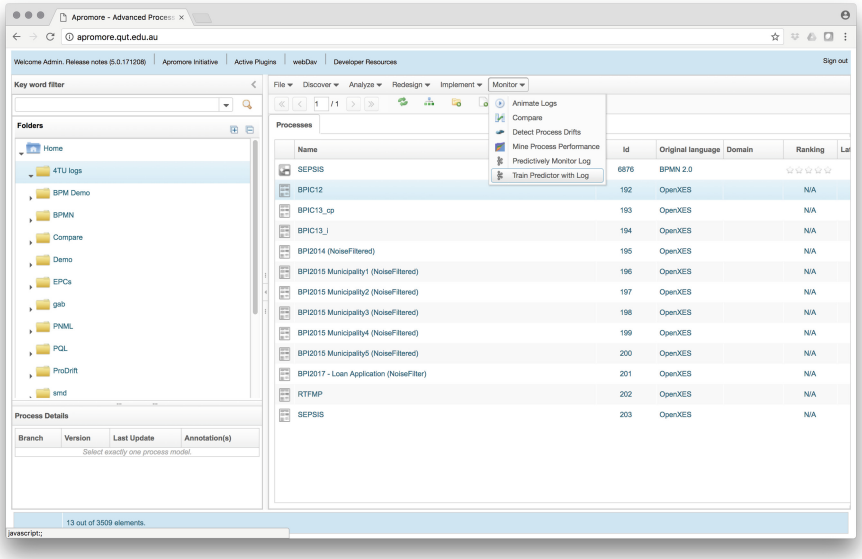


Fig. 2. Apromore's Portal with predictive monitoring functionality highlighted.

- *Late supply*. It is a boolean variable indicating whether or not the case will be closed before the supply date of that case.
- *Delay Rank* indicating the degree of potential delay, namely "Just in case", "Mild", "Moderate", "Severe".
- *Next activity* indicating which activity will be performed right after the current one.
- *Remaining time* until case completion.

For use with the Training plugin bundle, we extracted an event log of completed purchased orders, while ongoing orders were fed into the Runtime plugin bundle to make predictions for them. The log contains a number of case attributes and event attributes ready to be used to train the models. Furthermore, in order to make more accurate predictions, we performed some basic feature engineering before feeding the log into Apromore. For example, to take into account resource contention, we added the number of currently open cases as an event attribute.

4 Training Plugin Bundle

The Training plugin bundle provides several algorithms for generating predictive models suitable for different types of predictions. Specifically, it is able to build models for predicting remaining time, the next activity to be performed, whether a case will exceed a specified duration threshold, as well as various static case

attributes, for example, the total cost of the order. To this aim, the Training bundle involves two phases: a training and a validation phase. In the former, one or more predictive models are fitted; in the latter, their suitability to the specific dataset is evaluated, so as to support the user in selecting the predictive model that ensures the best results.

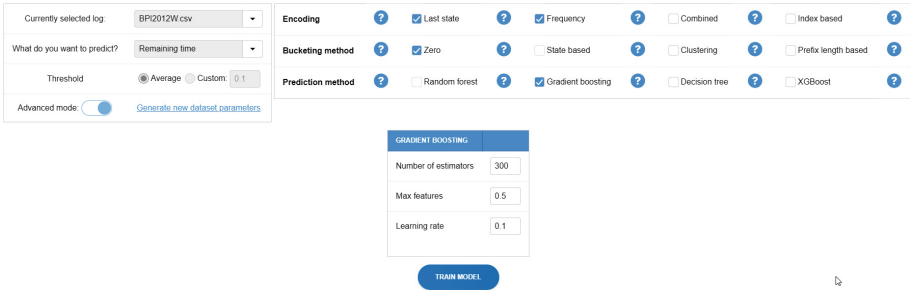


Fig. 3. Training configuration screen.

The Training bundle is composed of a front-end application (Fig. 3), which allows users to select the prediction methods and to assess the goodness-of-fit of the built models, and a back-end application for the actual training and validation. From the data flow perspective, the back-end application performs several tasks shown in Fig. 4.

Firstly, when a user uploads their log, the tool extracts and categorizes data attributes of the log into static case attributes and dynamic event attributes. On the other hand, each attribute needs to be designated as either numeric or categorical. These procedures are performed automatically upon the log uploading. Nevertheless, the user is given an option to override the automatic attribute definitions. Proper attribute categorization ensures best training data quality. The resulting definitions are saved in a configuration file in a JSON format (Fig. 5).

Secondly, the log is internally split into training and validation set in a 80-20 proportion. The former is used to train the model, while the latter is used to evaluate the predictive power of the model. Next, all traces of a business process need to be represented as fixed-size feature vectors in order to train a predictive model. To this end, several encoding techniques were proposed in [3] and further refined in [5], out of which we support four, namely last state encoding, frequency (aggregation) encoding, combined encoding and lossless index-based encoding.

While some of existing predictive process monitoring approaches train a single classifier on the whole event log, others employ a multi-classifier approach by dividing the prefix traces in the historical log into several buckets and fitting a separate classifier for each such bucket. At run-time, the most suitable bucket for the ongoing case is determined and the respective classifier is applied to make a prediction. Various bucketing types have been proposed and described in detail in [5]. The Training bundle supports four types of bucketing: zero bucketing (i.e.

fitting a single classifier), state-based bucketing, clustering-based bucketing and prefix length-based bucketing.

For each bucket of feature vectors, we train a predictive model using one of four supported machine learning techniques: decision tree, random forest, gradient boosting and extreme gradient boosting (XGBoost). For each technique, a user may manually enter the values of the most critical hyperparameters, e.g the number of weak learners (trees) in a random forest model.

In order to accommodate users with varying degrees of expertise in machine learning and predictive process monitoring, the plugin bundle offers two training modes – basic and advanced. By default, the basic mode is activated wherein a user only needs to choose the log and prediction target. If the prediction target is based on the logical rule – whether the case duration will exceed the specified threshold, a user is also invited to key in the threshold value. For all the other settings – bucketing method, encoding method and prediction method and its hyperparameters – the default values which usually achieve the best prediction accuracy will be used. Experienced users may switch the advanced mode toggle and manually choose bucketing, encoding and prediction method settings or any plausible combination thereof. The latter is especially useful when a user wants to train and compare multiple models, e.g. using various sequence encoding methods.

The status of the trained models can be verified using the collapsible drawer in the right-hand corner. Upon the training completion, a serialized Python object in the pickle format is produced. It describes a trained predictive model and includes:

- Configuration parameters of the predictors (whether it is a classifier or a regressor, what learning algorithm it uses).
- Definition of each column of the event log (static or dynamic, numeric or categorical). This information allows the Runtime plugin bundle to construct a feature vector from a given partial trace.

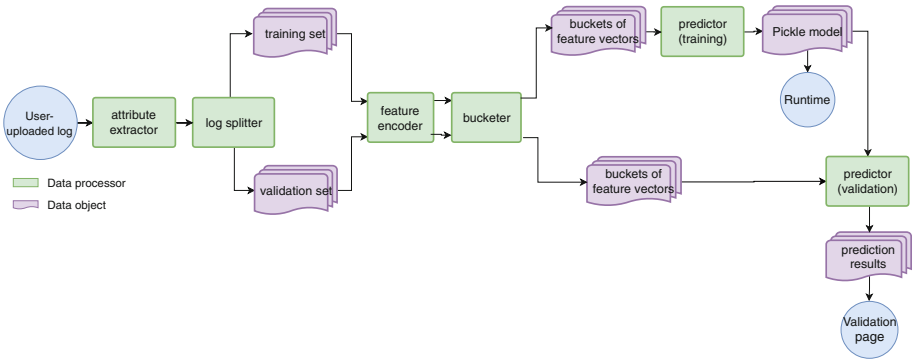


Fig. 4. High-level data flow diagram of the Training plugin bundle.

```
{
"case_id_col":"Line_ID",
"activity_col":"Activity_Name",
"timestamp_col":"Activity_End_Time",
"static_cat_cols":["Supplier_ID","Supplier_Location","
    Delivery_Type","Sector"],
"dynamic_cat_cols":["Employee_ID","weekday"],
"static_num_cols":["Line_Total_Cost","Target_Supply_Date_delta"]
    ,
"dynamic_num_cols":["elapsed","timesincelastevent","open_cases"]
    ,
"ignore":["Activity_Start_Time","hour","Milestone","Key","
    Target_Supply_Date","Part_ID"],
"future_values":["Late_Supply","Delay_Rank"],
}
```

Fig. 5. Example training configuration file.

- For each bucket, the trained model, ready to be taken as input by the selected prediction algorithm, e.g. in the case of decision trees, the whole tree representation.
- The bucketing function, which given an input sample, allows us to determine from which bucket a predictor should be taken.

The predictive power of the trained model(s) can be evaluated on a held-out validation set. By default, a user will see the average accuracy across all partial traces after a certain number of events have completed. This evaluation method was also used in [3,5]. For classification tasks (e.g. prediction of Late Supply and Delay Rank), a user can choose which metrics to plot among accuracy score, F1 score and logarithmic loss. For regression tasks (e.g. remaining time), a user can choose between mean absolute error and root mean square error, either raw or normalized. The accuracy of a particular model can be visually compared with that of other models trained for the same log and the same prediction target (Fig. 6). Additionally, one can check a scatter plot of predicted vs. actual values (for regression tasks) or a confusion matrix (for classification tasks) and assess the relative importance of each feature for the chosen predictor.

5 Runtime Plugin Bundle

Once the predictive models have been created, they can be deployed to the Runtime predictive monitoring environment of Apromore, to make predictions on ongoing cases. The Runtime plugin bundle can be used to stream an event log from the repository, or hook into an external stream. Either way, the input stream is transformed into a stream of predictions which is visualized in a Web-based dashboard. The transformation is implemented using the dataflow pipeline in Fig. 7.

The pipeline is built on top of the open-source Apache Kafka stream processing platform.[2] The "predictor" components of the pipeline are the predictive

[2] https://kafka.apache.org.

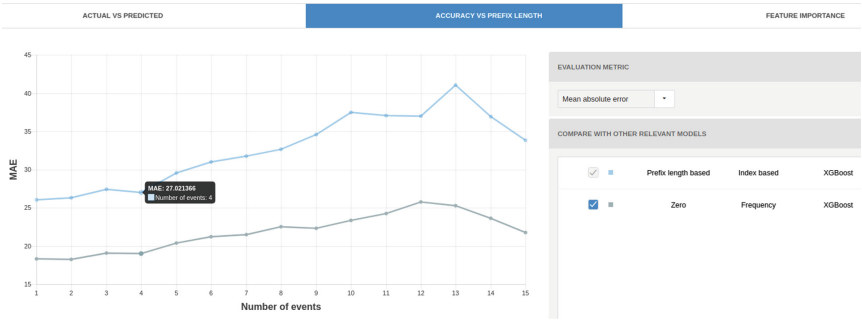


Fig. 6. Model validation page of the Training plugin bundle.

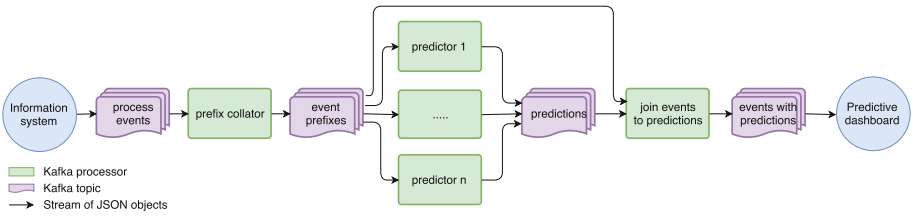


Fig. 7. High-level data flow diagram of the Runtime plugin bundle.

models from the Training plugin bundle. The "topic" components are network-accessible queues of JSON messages with publisher/subscriber support. This allows the computationally intense work of the predictors to be distributed across a cluster of networked computers, providing scalability and fault-tolerance. The "collator" component accumulates the sequence of events-to-date for each case, such that the prediction is a stateless function of the trained predictive model and of the case history. This statelessness is what allows the predictors to be freely duplicated and distributed. The "joiner" component composes the original events with the various predictions, ready for display on the dashboard.

The dashboard provides a list of both currently ongoing cases (colored in gray) as well as completed cases (colored in green), as shown in Fig. 8. For each case, it is also possible to visualize a range of summary statistics including the number of events in the case, its starting time and the time when the latest event in the case has occurred. For the ongoing cases, the Runtime plugin bundle provides the predicted values of the performance indicators the user wants to predict. For completed cases, instead, it shows the actual values of the indicators. In addition to the table view, the dashboard offers other visualization options, such as pie charts for case outcomes and bar charts for case durations. It is also possible to export the predictions in a CSV file, for periodic reporting and for importing into third-party business intelligence tools.

Process workers and operational managers can set some process performance targets and subscribe to a stream of warnings and alerts generated whenever

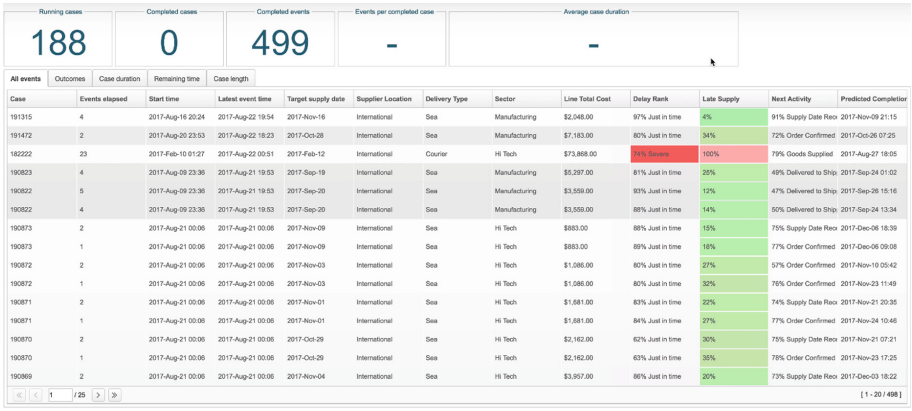


Fig. 8. Main view of the dashboard in the Runtime plugin bundle. (Color figure online)

these targets are predicted to be violated. Thus, they will be capable of making informed, data-driven decisions to get a better control of the process executions. This is especially beneficial for processes where process participants have more leeway to make corrective actions (for example, in a lead management process).

6 Conclusion

Through the integration with Nirdizati, Apromore offers a configurable full-stack Web tool that supports users in selecting and tuning various prediction models, and that enables the continuous prediction of different process performance indicators at runtime. Predictions can be presented visually in a dashboard or exported for periodic reporting.

Video demos of the model training and of the runtime functionality can be found at http://youtu.be/xOGckUxmrVQ and at http://youtu.be/Q4WVebqJzUI respectively. The source code is available under the LGPL version 3.0 license at https://github.com/apromore/ApromoreCode.

References

1. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, 2nd edn. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-56509-4
2. Jorbina, K., Rozumnyi, A., Verenich, I., Francescomarino, C.D., Dumas, M., Ghidini, C., Maggi, F.M., Rosa, M.L., Raboczi, S.: Nirdizati: a web-based tool for predictive process monitoring. In: Proceedings of the BPM Demo Track (2017)
3. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21

4. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Corts, A.: Predictive monitoring of business processes: a survey. IEEE Trans. Serv. Comput. **PP**(99), 1 (2017)
5. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. CoRR abs/1707.06766 (2017)
6. Verenich, I.: A general framework for predictive business process monitoring. In: Proceedings of CAiSE 2016 Doctoral Consortium co-located with 28th International Conference on Advanced Information Systems Engineering (2016)

Modelling Realistic User Behaviour in Information Systems Simulations as Fuzzing Aspects

Tom Wallis(✉) and Tim Storer

University of Glasgow, Glasgow, Scotland
w.wallis.1@research.gla.ac.uk, timothy.storer@glasgow.ac.uk

Abstract. In this paper we contend that the engineering of information systems is hampered by a paucity of tools to tractably model, simulate and predict the impact of *realistic* user behaviours on the emergent properties of the wider socio-technical system, evidenced by the plethora of case studies of system failure in the literature. We address this gap by presenting a novel approach that models ideal user behaviour as workflows, and introduces irregularities in that behaviour as aspects which fuzz the model. We demonstrate the success of this approach through a case study of software development workflows, showing that the introduction of realistic user behaviour to idealised workflows better simulates outcomes reported in the empirical software engineering literature.

1 Introduction

Information systems are operated within a wider organisational context, characterised by the needs, demands and behaviours of individual users, interpersonal relationships, organisational structures, business processes, legal and regulatory standards, and cultural norms [2,21]. The influence of this *socio-technical* interplay on system behaviour (and often failure) has been described in multiple and diverse case studies of information systems, including automated emergency vehicle dispatch [22], electronic voting [17] and stock market trading systems [26]. In each case, system failure cannot be attributed to either purely user behaviour or the information system(s), but instead to an interplay between both factors within the overall socio-technical system.

The contention in this paper is that these failures arise because systems engineers lack the tools and methods to efficiently model and accurately simulate the interaction between the information systems and their organisational context, which comprise a socio-technical system. Without such facilities, systems engineers cannot predict potential socio-technical system behaviour during information system design. Simulating this interaction between the information system and users is hard because user behaviour is heterogeneous, contingent and evolutionary, leading to irregularities in workflows envisaged by systems engineers. Different users have different abilities, training and experiences and can experience phenomena such as distraction, misjudgements, exhaustion and

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 254–268, 2018.
https://doi.org/10.1007/978-3-319-92901-9_22

confusion that can have significant influence on how and when a user completes a task. For example, a novice developer working within a large software team may be unaware of software development best practices described in a workflow, perhaps forgetting to make frequent commits to a version control server. Conversely, an experienced developer may omit steps that they consider unnecessary, such as peer reviews of their code, to optimise their performance.

Information system users may also adapt their behaviour due to contingencies that arise from faults in the information system or the behaviour of other users in the environment [23]. Continuing the example scenario, a software development team may be tempted to begin reducing quality assurance efforts as a deadline for a release approaches, in an effort to ensure all required features are completed [3]. Behaviour is also continually evolving, as the users of a system adapt to new circumstances, discover optimizations to their workflows, adapt the workflow to suit local organisational priorities or take shortcuts [6]. In the example scenario, it is reasonable to anticipate that a novice developer's behaviour will gradually evolve into that of an expert as they gain more experience. As a consequence, the de facto behaviour exhibited within a system may differ from that envisaged by system architects in idealised workflows.

Despite the potential impact on system performance, the complexity of user behaviour is cumbersome to model and therefore predict using conventional systems engineering notations for describing workflows, such as BPMN [20], activity diagrams [19] and YAWL [14]. Attempting to model all the potential sequences of actions that result from realistic user behaviour using these approaches inevitably results in models that are either too abstract or narrow to be informative, or too complex to be tractable. Approaches that abstract the complexity of user behaviour, such as i* [29], Kaos [28] and Responsibility Modelling [23] lack sufficient detail to generate executable simulations.

The key insight of this paper is that the same behavioural irregularities can affect many different idealised workflows. Conversely, a given user's behaviour can be modelled as the combination of expected ideal behaviour and the irregularities that affect how that user exhibits their behaviour. Therefore, *we propose and demonstrate the separate modelling of behavioural irregularities from workflows themselves, showing that they can be represented as cross-cutting concerns which can be applied to an idealised workflow model.*

The two contributions of this paper are as follows:

- A method for simulating user interactions with information systems that allows for the separate modelling of idealised descriptions of workflows and the effects of irregularities on those workflows. The approach is implemented in a framework comprising of aspect oriented programming [11] and dynamic code fuzzing [25]. This combination allows for irregularities to be introduced into workflow models without sacrificing readability or simplicity.
- A demonstration of the efficacy of the approach in a case study comparison of Test Driven Development (TDD) and Waterfall software development workflows. The case study demonstrates that models of idealised behaviour in software development do not conform with the empirical evidence which

suggests that TDD outperforms Waterfall [5,12,15], but that models which incorporate realistic user behaviour when interacting with information systems do.

The rest of this paper is structured as follows. Section 2 discusses related work, covering existing techniques for modelling socio-technical workflows and the limitations encountered in the literature. Section 3 introduces the case study problem domain selected to evaluate our approach, and presents the method for constructing models of information systems and associated workflows. Section 4 describes the development of aspect oriented fuzzing of user behaviour and the software tool developed for this purpose. Section 5 presents our evaluation of the case study and Sect. 6 discusses conclusions and future work.

2 Related Work

This section presents a literature review of approaches to modelling the interaction between users and information systems. Graphical notations have received considerable attention, perhaps due to their perceived efficacy in communicating specifications between users, customers and system architects. Workflow description languages, such as UML activity diagrams [19], BPMN [20] and YAWL [14] can be used to denote workflows visually as directed graphs composed of activities, as well as forking and merging workflows. Although activity diagrams are more widely adopted than BPMN, the latter provides a richer modelling notation, allowing for discrimination between tasks, activities and transactions; triggering and orchestrating concurrent activities using messages; the identification of information resources need to realise an activity; and the orchestration of activities across organisational boundaries. YAWL also provides for a richer range of workflow requirements than activity diagrams, including sophisticated forking and merging rules, separation between workflow specifications and executions and resourcing and data requirements.

Describing realistic user behaviour in workflow notations can be difficult, because of the basic assumption that all paths in a workflow can be completely described at a given level of granularity, and that more complex details can be encapsulated within coarser grained modules. As argued in Sect. 1, user behaviours are inherently complex, making such refinement based techniques difficult to apply. As Israilidis et al. [16] have argued, the 'unknowns' in a socio-technical system may be far more significant than the 'knowns'.

Several authors have therefore discussed alternative techniques for modelling socio-technical systems with support for complexity in behaviour. Both i* [29] and KaOS [8] are goal oriented notations for modelling socio-technical systems [28]. In contrast to workflows, goal oriented approaches primarily capture what system actors are seeking to achieve. Goals can be de-composed into a sub-goal hierarchy using logical operators to express the form of decomposition. Goals can also be annotated with strategies and/or resource requirements to support automated analysis. Yu [29] argued that socio-technical systems should

be viewed as collections of collaborating actors, each with their own (potentially conflicting) objectives. Eliciting and analysing the actors' intents allows the inter-dependencies between actors and the overall behaviour of the system to be understood, without the need for explicit models of individual workflows. Herrmann and Loser [13] introduced techniques for annotating goal oriented system models with vagueness. The annotation allows for the distinction between consistent vagueness (due to abstraction) and inconsistent vagueness due to omission. In principle, this approach allows for the identification of aspects of a model that may be subject to irregularity. However, the notation is not accompanied by a formal semantics, or other means of supporting automated analysis.

Other authors have extended goal oriented approaches to provide greater flexibility. Sommerville et al. [23] argued that stakeholders often struggle to express their behaviour within a socio-technical system in terms of goals. Instead, they argue that the concept of responsibilities, the duties held by an actor in a system, are a more intuitive means of describing system behaviours that also capture a variety of contingencies. Various notational features have been provided for extending responsibility modelling, including for linking between responsibilities and required resources and for associating indicative workflows [10]. However, these are not accompanied with a means of evaluating the effects.

The contributions made by Herrmann and Loser [13] and Sommerville et al. [23] demonstrate the recognition of the need for methods which capture the complexity of user interactions with information systems. However, to the best of our knowledge there has been no attempt to go further and develop methods which can simulate the effects of this behaviour on the overall system. The observation made in Sect. 1 that the causes of irregularities in workflow execution are separate concerns from the model of the workflow itself invites a novel approach to the problem of simulating realistic user behaviour. This approach combines dynamic software fuzzing, which provides variation in the workflow model, with aspect oriented programming (AOP), to decouple this behavioural variation from the idealised model.

The representation of cross-cutting concerns, such as access control, logging and concurrency as program aspects has been extensively studied in the literature [1]; and the use of code fuzzing techniques have been employed extensively in quality assurance practices, to assess the impact of unexpected events on software systems, including fuzzing of software configurations at runtime [7], mutation testing of test suites [9] and fuzzing of inputs test data to search for software security vulnerabilities [25]. However, we have not discovered any similar work on the application of fuzzing techniques to dynamically inject variation into a simulation model at runtime in order to mimic complex user behaviour. Similarly, we are unaware of any research that utilises AOP techniques in simulation models, either for socio-technical user behaviours or elsewhere.

3 Team Based Software Development Case Study

In this section we demonstrate our approach to modelling information systems and associated idealised workflows in socio-technical systems through a case

study. We have chosen a case study of team based software development, in which we will explore the efficacy of the workflows of two software development lifecycles (SDLC): Waterfall and Test Driven Development (TDD). The case study was chosen as representative of a socio-technical system, combining different user roles (developer, project manager), information systems (version control server and client, and the software development effort itself) and a variety of well documented idealised workflows (implementation, testing, debugging etc.). Further, there is a growing consensus amongst software development professionals and in the academic literature that TDD is a more resilient SDLC than Waterfall to realistic human behaviour [5,12,15]. The case study therefore provides an opportunity to test whether the modelling approach can be used to distinguish between the effects of idealised and realistic user behaviours in a socio-technical system. The model is implemented in an agent oriented framework, Theatre_Ag, written in Python. Fragments of the case study are provided as examples, but the full source code is also available for inspection [24].

The first step in the modelling approach is to represent the state and functions of the information systems in the case study. The domain is modelled as a collection of Python classes, as shown in Fig. 1, following an object oriented approach [19]. The core information system is a version control server (VCS) which hosts the software development effort. Software developers interact with the server via a VCS client. Both the server and clients maintain copies of the software system, which must be coordinated through a VCS update, conflict-resolve, commit cycle. Software systems are composed of features, representing user-facing specifications of the system's functionality. Features may be of varying size, requiring more or fewer code chunks to be implemented in order to be operational. Chunks may have dependencies on other chunks in the system. Each time a new chunk is added to a feature other chunks may also need to be modified, potentially creating further dependencies between chunks or introducing bugs. Features can be operated, causing some of the underlying chunks and any dependencies to also be operated. Bugs may become manifest during chunk operation, or through the execution of tests. Features can be debugged, resulting in the removal of detected bugs. Consequently, the more tests created for a feature, the greater the probability of detecting a given bug, easing the process of debugging. All of the information system behaviours described above and shown in the diagram are implemented as methods in Python classes.

The second modelling step is to define user behaviour workflows that will be followed by users, represented by agents in the simulation environment. These are also implemented as Python classes, collating related tasks to operate on a common state (the information systems). Figure 2 shows classes for two of the workflows created for the case study: change management and implementation. The state for the change management workflow is the VCS server and a client that manages the working copy. Separate task methods are provided for checking out the client, committing working copy changes to the server and resolving conflicts that arise during a commit task.

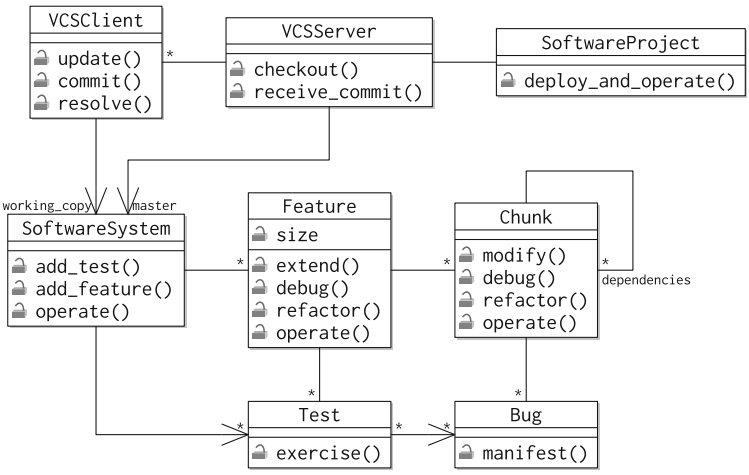


Fig. 1. The software development case study domain model, showing information system artefacts and functions that can be accessed by user workflows.

Note that many of the functions of the information system have side effects that are modelled stochastically, such as the possibility that an update from the version control server will result in conflicts with another client's commit. The workflows therefore explicitly pass a source of randomness to the information system model when these actions are invoked.

Further workflows were implemented for common software development activities, including the specification and implementation of features in the system, debugging and refactoring (reducing dependencies), and testing. Since workflows are modular they can also be organised hierarchically. Figure 2 shows how this modularity is exploited for the implementation workflow. The other workflows for modifying the structure of a software system (for adding chunks or tests and so on) also depend on the change management workflow in order to coordinate the distribution of changes within a team.

Two further workflows, Waterfall and Test Driven Development (TDD) were implemented to investigate the performance of different team based software development lifecycles (SDLC). The Waterfall SDLC [4] describes a linear staged approach to system development, in which specification of all features is followed by system implementation, testing, debugging, refactoring and final deployment. Conversely, TDD [3] prescribes the definition of tests for each feature, before proceeding to implementation and refactoring. Advocates of TDD argue that such an approach results in software that is delivered quicker and of higher quality because tests are explicitly linked to the specification and features are not committed without a passing test suite.

Once the models of the information systems and workflows are complete, simulations can be executed in Theatre_Ag, which are initiated by allocating a set of tasks (individual methods in a workflow) to agents in the system. This

```
class ChangeManagement(object):
    def __init__(self, vcs_server):
        self.vcs_server = vcs_server
        self.vcs_client = None

    @default_cost(1)
    def resolve(self, conflict, rand):
        self.vcs_client.resolve(conflict, rand)

    @default_cost(0)
    def commit_changes(self, rand):
        while True:
            try:
                self.vcs_client.commit()
                self.vcs_client.update(rand)
                break
            except CentralisedVCSException:
                self.vcs_client.update(rand)
                for conflict in self.vcs_client.conflicts:
                    self.resolve(conflict, rand)

    @default_cost(0)
    def checkout(self):
        self.vcs_client = self.vcs_server.checkout()


class Implementation(object):
    def __init__(self, change_management):
        self.change_management = change_management

    @property
    def working_copy(self):
        return self.change_management.vcs_client.working_copy

    @default_cost(1)
    def add_chunk(self, chunk_logical_name, feature, rand):
        feature.extend(chunk_logical_name, rand)

    @default_cost()
    def implement_feature(self, logical_name, rand):
        self.change_management.checkout()

        feature = self.working_copy.get_feature(logical_name)

        while not feature.is_implemented:
            self.add_chunk(len(feature.chunks), feature, rand)
            self.change_management.commit_changes(rand)

    @default_cost()
    def implement_system(self, rand):
        self.change_management.checkout()
        for feature in self.working_copy:
            self.implement_feature(feature, rand)
```

Fig. 2. Python code for workflows describing change management and software implementation.

may lead to the creation and execution of further tasks by agents, or to the allocation of tasks to other agents. In the case study, simulations were configured to represent a team of software developers working on a software project following both types of SDLC. For Waterfall, the initial configuration specified an agent in the team with the role of project manager. This agent is directed to

allocate tasks for each stage of the software development process (specification, implementation, testing, debugging, refactoring) to other agents in the team and waits for all tasks in each stage of the development process to be complete before allocating tasks in the next stage. For TDD, all members of the software development were configured to draw features to be implemented from the project backlog and implement them following TDD.

All task execution is coordinated relative to a single simulation clock, enabling both the explicit representation of problem domain time for controlling task execution and concurrent execution of these tasks by agents. Task costs are denoted using the `@default_cost` decorator in a workflow as shown in Fig. 2. The agent pauses execution of a task until sufficient ticks from the clock have been received and then resumes execution. This mechanism is implemented transparently with respect to workflow classes, easing the modelling burden. The execution of tasks and their duration is logged in a tree-like structure by each agent, supporting later inspection and analysis of behaviour.

4 Modelling Irregularity in Behaviour as Fuzzing Aspects

In this section the approach to modelling irregularities in user behaviour is demonstrated. To achieve this, an irregularity is modelled as a *dynamic fuzzing aspect* using the library developed for this purpose, PyDySoFu [27]. PyDySoFu interrupts the invocation of workflow task methods in the background during the execution of a simulation. When a workflow task method that can be subject to irregularity is invoked, PyDySoFu pauses the simulation thread of execution and passes the method context and structure (represented as the abstract syntax trees of the statements in the method body) to a modeller defined fuzzing function. The fuzzing function then manipulates the structure of the method AST and context as desired and returns the AST to PyDySoFu. In a final stage, PyDySoFu compiles the altered AST and resumes simulation execution by invoking the altered method. Further implementation details are omitted for brevity, but the full source code of the PyDySoFu library and documentation is available for inspection [27].

In order to evaluate the feasibility of the approach, a fuzzing aspect that mimics *distraction* was implemented using PyDySoFu. The distraction aspect represents irregularities in expected behaviour when a user engaged in one task becomes distracted by another, irrelevant activity and so does not complete later steps in the intended workflow. The probability of distraction increases as the simulation proceeds, i.e. the probability of moving to an irrelevant task increases the longer a simulation has been executing. For example, a software development team may be distracted from the implementation of tests and debugging as the pressure to complete a set of features increases, due to an impending release deadline. Conversely, the aspect also allows the probability of distraction to be reduced by increasing the concentration of a simulated user (i.e. users with better concentration do not get distracted as easily). The implementation of the `incomplete_procedure` fuzzing aspect is shown in Fig. 3.

```
def default_pmf(concentration=1.0):

  def _probability_distribution(remaining_time, probability):
    adjusted_probability =\
          probability ** ((remaining_time + 1.0) * concentration)

    return sys.maxint if adjusted_probability == 1.0 \
        else int(1.0 / (1.0 - adjusted_probability) - 1)

  return _probability_distribution

def incomplete_procedure(random, pmf=default_pmf()):

  def _incomplete_procedure(steps, context):
    clock = context.agent.clock
    remaining_time = clock.max_ticks - clock.current_tick

    n = pmf(remaining_time, random.uniform(0.0, 0.9999))

    fuzzer =\
      recurse_into_nested_steps(
        target_structures={ast.While, ast.For, ast.TryExcept),
        fuzzer=filter_steps(
          choose_last_steps(n, reapply=False),
          replace_steps_with(
            replacement='self.agent.idling.idle()'
      )))
    return fuzzer(steps, context)

  return _incomplete_procedure
```

Fig. 3. An aspect which dynamically truncates the execution of a workflow to represent a distracted user operating a system.

The fuzzing aspect is parameterised, so the actual aspect is defined as the inner function `_incomplete_procedure`. The first four lines of the inner function configures a probability mass function (PMF) for choosing how many steps should be removed by the aspect, `n`, based on the remaining time in the simulation and the agent's concentration. The default PMF is shown in the figure, although this can be configured by the modeller.

PyDySoFu is accompanied by a library of aspects from which more complex aspects can be constructed and parameterised. This includes functions for recursing into control structures and selecting and replacing steps based on their position in the AST. These features are exploited in the second part of the fuzzer, which recurses into any nested control structures (while, for and try) found in the method and applies a filter to the bodies of these statements, tail first. The `filter_steps` aspect selects up to `n` steps to remove from the body and replaces each of them with an invocation of `idle()`, causing the agent executing the workflow to wait one tick for each replaced step. The `filter_steps` aspect is then applied successively up through the target AST. The `choose_last_steps` filter is configured to track how many steps it has already removed, so that the value of `n` is calculated across the entire body of the fuzzed task method, rather than at each level in the recursion.

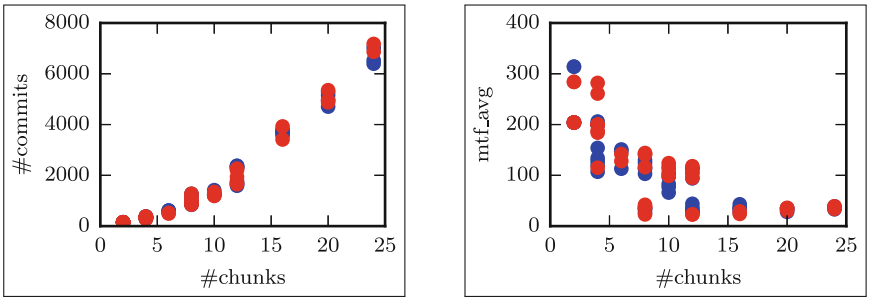
Notice that the fuzzing aspect models the effects of distraction *independently* of its application to a particular workflow. This allows the effects of this irregular behaviour to be applied transparently to any of the different user workflows in the simulation. In the current work, it is not claimed that the implementation of the aspect is empirically accurate, rather that it is representative of a qualitatively recognisable phenomena that is better modelled separately from the workflow; and that the approach allows for the effect of the phenomena on a workflow at different degrees of severity to be assessed. Also, it is not claimed that distraction is the *only* cause of irregularities in behaviour in software development workflows. Rather, we present distraction as an example of a cause of irregularity that may affect many different workflows in the same way, and demonstrate the approach to modelling it separately as a cross-cutting concern.

5 Evaluation

This section presents the results of applying the user behaviour fuzzing method to the case study. Evaluating the correctness of simulation techniques intended for large scale systems is notoriously difficult, since the usual rationale for developing the simulation is the lack of available empirical observations of the real world phenomena [18]. However, in the context of the case study, limited empirical evidence exists that suggests TDD outperforms Waterfall in terms of features delivered and software quality [5,12,15]. The evaluation will therefore adopt a hybrid strategy.

Following Naylor and Finger [18], the model of the problem domain will first be tested for *plausibility*. A comparison of simulations of user behaviour with and without the application of dynamic aspect fuzzing will then be made. The first comparison will test whether two SDLC workflow models perform equivalently when executed in ideal circumstances, i.e. that simulations of ideal behaviour do not conform with results presented in the literature. The second comparison will test whether the TDD workflow simulation out-performs Waterfall when user behaviour varies from the ideal workflow due to the effects of distraction, in conformance with the literature.

Simulations of the problem domain were configured as follows. A single source of randomness was initialised for each configuration with a constant seed to provide for repeatability of the experiment and to provide for comparison between configurations. All configurations were executed for a team of 3 software developers, with simulations executing for a maximum of 500 ticks. Each configuration was run for 25 different projects. The system 'built' by the simulation was operated 10 times, in order to measure software quality. Operation of the simulated system entails random selection and operation of features up to 750 times. If a bug is manifested during operation then the system operation halts. The length of the average trace of the operations of the system was recorded as the system's mean time to failure (MTF). Software projects consisted of up between 1 and 6 features, with projects divided into small (2 chunks per feature) and large (4 chunks per feature). Both the Waterfall and TDD SDLC workflows were simulated. In either case, the incomplete procedure distraction aspect was applied to



(a) Commits against project size.

(b) Mean time to failure.

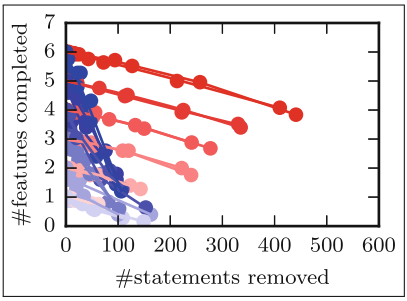
Fig. 4. Scatter plots of software development simulations without fuzzing for Waterfall (blue) and TDD (red) projects. (Color figure online)

either the high level software development workflow, or to the set of lower level development activity workflows (implementation, change management etc.). The aspect was applied using concentration values between 0.001 and 5.0. Raw simulations results are available for inspection [24].

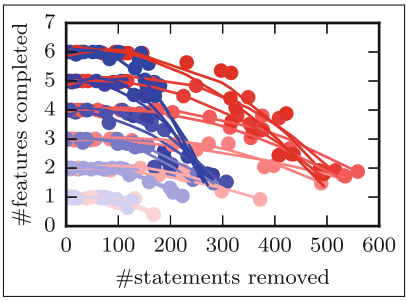
In the first part of the evaluation, simulations of the software development workflow were considered for ideal behaviours, without the effects of distraction applied. Figure 4 illustrates two comparisons of simulations to establish problem domain plausibility. Figure 4a shows that there is an exponential increase in the number of commits to the version control server as project size increases. This result is to be expected due to the exponential increase in the number of additional modifications, tests, debugging and refactoring activities that would be required as a project grows in scale. Similarly, Fig. 4b shows a decline in project quality (a reduction in the MTF from 300 to just above 0) as project size increases. Again, the additional complexity of larger projects increases the probability of a bug causing a system failure. These results strengthen the claim that the model of the problem domain is plausible.

A further observation from the figures is that TDD and Waterfall workflows perform equivalently when workflows are not subject to user distraction. When ideal workflows are executed for both Waterfall (blue) and TDD (red), the results show similar metrics for each. This confirms the expectation that the simulations of idealised workflows, while behaving plausibly, do not conform with expectations from the literature that TDD should outperform Waterfall.

We now proceed to compare the performance of the two strategies when subject to behavioural irregularity caused by distraction. Figure 5 shows scatter plots for the effect of distraction for subsets of workflows on productivity. In both plots, the number of features implemented is plotted against the total number of workflow statements removed by fuzzing (Waterfall in blue, TDD in red). Figure 5a shows the effect when only the high level software development workflow (Waterfall or TDD) is fuzzed, whereas Fig. 5b shows the effect of fuzzing lower level activities (change management, implementation, testing, debugging



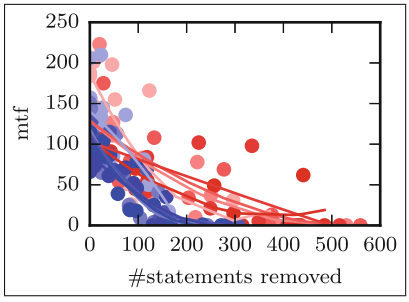
(a) Fuzzing of Waterfall and TDD.



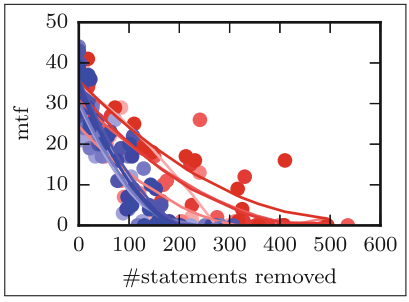
(b) Fuzzing of low-level workflows.

Fig. 5. Scatter plots and trend lines of feature completion for Waterfall (blue) and TDD (red) for distraction applied to selected workflows. (Color figure online)

and refactoring). Both figures clearly demonstrate that increasing distraction reduces the number of completed features. However, in both configurations, TDD is more resilient to distraction fuzzing than Waterfall, with a larger proportion of features implemented as steps are removed. In Fig. 5, both SDLC workflows appear to be largely immune to distraction until around 100 steps are removed, after which feature completion declines, with steeper declines for Waterfall.



(a) Small projects.



(b) Large projects.

Fig. 6. Scatter plots and trend lines of MTF for Waterfall (blue) and TDD (red) for distraction applied to all workflow combinations. (Color figure online)

Figure 6 shows the effect of distraction on MTF, again distinguishing between Waterfall (blue) and TDD (red) SDLCs. In these plots, both workflow fuzzing configurations described above are plotted together, as their effects are similar. Instead, small and large feature projects are plotted separately, as these have different orders of magnitude MTF. Similar to the plots of feature completion, the plots of MTF suggest that TDD is more resilient to fuzzing than Waterfall, which experiences a much more dramatic decline as more statements are removed. In

summary, the two comparisons of productivity and quality demonstrate that the application of distraction fuzzing results in more realistic simulations of user behaviour in software development workflows, as found in the available literature.

6 Conclusions

This paper has presented and evaluated a novel approach to modelling and simulating realistic user behaviours when interacting with information systems, through the use of model fuzzing aspects. The novelty of this approach is the ability to model the causes of realistic user behaviour (such as distraction) separately as aspects that can be applied flexibly to many different workflows in a simulation. The efficacy of the approach is demonstrated in the introduction of realistic user behaviour to an example system simulation, which better conforms with results in the literature without altering the original workflows.

This proof of concept presents the opportunity for a substantial research agenda in the modelling and simulation of user interaction with information systems, in order to better predict emergent system properties. Further research is necessary to test the viability of the approach in more complex, large scale simulations, such as the election e-counting system described by Lock et al. [17], comprising whole information infrastructures and many diverse users. This will provide opportunities to experiment with a variety of other causes of irregularity in user behaviour, such as subjective misjudgements, exhaustion, reordering of steps and shortcut taking. Interdisciplinary research is required to develop and validate the realism of the fuzzing aspects that implement these user behaviours, and a library of pre-validated aspects could be made available to ease modelling. There is a need to link empirical evidence of the causes and occurrences of these behaviours to their impact on a workflow's simulation and real-world execution.

The overall aim of this research agenda is towards simulation techniques that can have predictive capabilities suitable for informing systems engineering decisions, before resources are allocated toward them. Such tools would do much to progress the current craft of large scale information systems engineering.

Acknowledgements. The authors would especially like to thank Gordon Baxter, Phil Trinder and Russell Lock for their help when revising this paper, and OBASHI Technology for helping to fund the work.

References

1. Ali, M.S., Babar, M.A., Chen, L., Stol, K.: A systematic review of comparative evidence of aspect-oriented programming. Inf. Softw. Technol. **52**(9), 871–887 (2010)
2. Bade, D.: Structures, standards, and the people who make them meaningful. In: Structures and Standards for Bibliographic Data (Chicago, IL), May 2007
3. Beck, K.: Test Driven Development by Example. Signature. Addison Wesley, Boston (2002)

4. Benington, H.D.: Production of large computer programs. Ann. Hist. Comput. **5**(4), 350–361 (1983)
5. Bhat, T., Nagappan, N.: Evaluating the efficacy of test-driven development: industrial case studies. In: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, pp. 356–363. ACM (2006)
6. Bonen, Z.: Evolutionary behavior of complex sociotechnical systems. Working Paper 1056–79, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Institute of Technology, 50 Memorial Drive, Cambridge, Massachusetts 02139, April 1979
7. Dai, H., Murphy, C., Kaiser, G.: Configuration fuzzing for software vulnerability detection. In: International Conference on Availability, Reliability, and Security, ARES 2010, pp. 525–530. IEEE (2010)
8. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. **20**, 3–50 (1993)
9. DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on test data selection: help for the practicing programmer. IEEE Comput. **11**(4), 34–41 (1978)
10. Dewsbury, G., Dobson, J. (eds.): Responsibility and Dependable Systems. Springer, London (2007). https://doi.org/10.1007/978-1-84628-626-1
11. Filman, R.E., Friedman, D.P.: Aspect-oriented programming is quantification and obliviousness. Technical report 01.12, Research Institute for Advanced Computer Science, Indianna University, Bloomington, October 2001. Presented at the Workshop on Advanced Separation of Concerns, OOPSLA 2001
12. George, B., Williams, L.: A structured experiment of test-driven development. Inf. Softw. Technol. **46**(5), 337–342 (2004)
13. Herrmann, T., Loser, K.U.: Vagueness in models of socio-technical systems. Behav. Inf. Technol. **18**(5), 313–323 (1999)
14. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N. (eds.): Modern Business Process Automation - YAWL and its Support Environment. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-03121-2
15. Huang, L., Holcombe, M.: Empirical investigation towards the effectiveness of test first programming. Inf. Softw. Technol. **51**(1), 182–194 (2009)
16. Israilidis, J., Lock, R., Cooke, L.: Ignorance management. Manag. Dyn. Knowl. Econ. **1**(1), 71–85 (2013)
17. Lock, R., Storer, T., Harvey, N., Hughes, C., Sommerville, I.: Observations of the Scottish elections 2007. In: eGov 2007, Third e-Government Workshop, Leeds, UK, September 2007. http://www.indeedproject.ac.uk/publications/pdfs/lock07observations.pdf
18. Naylor, T., Finger, J.: Verification of computer simulation models. Manag. Sci. **14**(2) (1967)
19. OMG: OMG Unified Modeling Language (OMG UML) Superstructure. Object Management Group, v2.1.2 edn, November 2007
20. OMG: OMG Business Process Model and Notation (OMG UML) Superstructure. Object Management Group, 2.0 edn, January 2011
21. Pentland, B.T., Feldman, M.S.: Organizational routines as a unit of analysis. Ind. Corp. Change **14**(5), 793–815 (2005)
22. Robinson, B.: Limited horizons, limited influence: information technology the London ambulance service. In: Proceedings of the International Symposium on Technology and Society Technical Expertise and Public Decisions, pp. 506–514. IEEE Computer Society Press, June 1996

23. Sommerville, I., Lock, R., Storer, T., Dobson, J.: Deriving information requirements from responsibility models. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 515–529. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02144-2_40
24. Storer, T.: Software development workflow github repository, April 2017. http://www.github.com/twsswt/softdev-workflow
25. Takanen, A., DeMott, J.D., Mille, C.: Fuzzing for Software Security Testing and Quality Assurance. Information Security and Privacy Series. Artech House, Norwood (2008)
26. U.S. CFTC/SEC: Findings regarding the market events of May 6, 2010. Report of the staffs of the CFTC and SEC to the joint advisory committee on emerging regulatory issues. U.S. Commodity Futures Trading Commission. Three Lafayette Centre, 1155 21st Street, NW Washington, D.C. 20581. U.S. Securities & Exchange Commission 100 F Street, NE Washington, D.C. 20549, September 2010
27. Wallis, T., Storer, T.: Pydysofu github repository, April 2017. http://www.github.com/twsswt/pydysofu
28. Werneck, V.M.B., de Pádua Albuquerque Oliveira, A., do Prado Leite, J.C.S.: Comparing GORE frameworks: I-star and KAOS. In: Ayala, C.P., Silva, C.T.L.L., Astudillo, H. (eds.) Anais do WER09 - Workshop em Engenharia de Requisitos, Valparaíso, Chile, 16–17 Julho 2009 (2009)
29. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto (1995)

Author Index