Nico Herzberg, Matthias Kunze (Eds.)

Services and their Composition

6th Central European Workshop, ZEUS 2014
Potsdam, Germany, 20–21 February 2014
Proceedings

Volume Editors

Nico Herzberg
Universität Potsdam, Hasso-Plattner-Institut für Softwaresystemtechnik
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany
nico.herzberg@hpi.uni-potsdam.de

Matthias Kunze
Universität Potsdam, Hasso-Plattner-Institut für Softwaresystemtechnik
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany
matthias.kunze@hpi.uni-potsdam.de

Preface

In February 2014, we had the pleasure to host the 6th edition of the Central European Workshop on Services and their Composition (ZEUS) in Potsdam. This successful workshops series offers young research talents the chance to present and discuss early ideas and incomplete results, and to establish contacts between young researchers. 29 researchers and practitioners from Austria and Germany joined the workshop and contributed to the discussions of the presented work.

From all submissions, we selected 13 papers for the workshop program and proceedings. The program committee that evaluated all submissions for their relevance and scientific quality consisted of members of academia and industry. The accepted papers focused on topics in the area of Business Process Management and Service-oriented Computing, in particular process analysis, process modeling and improvement, flexible processes, process execution, and process data.

The workshop program has been completed by a keynote given by Ulf Leser from Humboldt-Universität zu Berlin, entitled "Research Opportunities in Scientific Workflows". Ulf Leser elaborated on the challenges of handling enormous amounts of data that are processed in scientific workflows. He pointed out that the rapid evolution of processing units requires flexible and robust execution environments.

Potsdam, March 2014

Nico Herzberg
Matthias Kunze

Organization

ZEUS 2014 is organized by the Business Process Technology group at the Hasso Plattner Institute (University of Potsdam), Potsdam, Germany.

Steering Committee

Oliver Kopp	University of Stuttgart, Germany
Niels Lohmann	University of Rostock, Germany
Karsten Wolf	University of Rostock, Germany

Organization, Program Committee Chairs

Nico Herzberg	Universität Potsdam, Germany
Matthias Kunze	Universität Potsdam, Germany

Program Committee

Rafael Accorsi	University of Freiburg, Germany
Stefan Appel	Technische Universität Darmstadt, Germany
Anne Baumgraß	Universität Potsdam, Germany
Dirk Fahland	Technische Universiteit Eindhoven, The Netherlands
Christian Gierds	Humboldt-Universität zu Berlin, Germany
Georg Grossmann	University of South Australia
Thomas Heinze	Friedrich-Schiller-Universität Jena, Germany
Nico Herzberg	Universität Potsdam, Germany
Meiko Jensen	Independent Centre for Privacy and Data Protection Schleswig-Holstein, Germany
Oliver Kopp	Universität Stuttgart, Germany
Agnes Koschmider	Karlsruher Institut für Technologie, Germany
Matthias Kunze	Universität Potsdam, Germany
Philipp Leitner	Vienna University of Technology, Austria
Henrik Leopold	Humboldt-Universität zu Berlin, Germany
Niels Lohmann	Universität Rostock, Germany
André Ludwig	Universität Leipzig, Germany
Jan Mendling	Wirtschaftsuniversität Wien, Austria
Andreas Meyer	Universität Potsdam, Germany
Stephan Reiff-Marganiec	University of Leicester, United Kingdom
Andreas Schönberger	Universtität Bamberg, Germany
Jan Sürmeli	Humboldt-Universität zu Berlin, Germany

Ruben Verborgh	Universiteit Gent, Belgium
Matthias Weidlich	Imperial College London, United Kingdom
Marco Zapletal	Vienna University of Technology, Austria

Additional Reviewers

Tobias Binz
Uwe Breitenbücher
Katharina Görlach
Christoph Gröger
Sebastian Wagner

Sponsoring Institutions

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH, Potsdam, Germany
Signavio GmbH, Potsdam, Germany

Contents

"BPELanon": Anonymizing BPEL Processes

Marigianna Skouradaki[1], Dieter Roller[1], Cesare Pautasso[2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
{skouradaki,dieter.h.roller,leymann}@iaas.uni-stuttgart.de
[2] Faculty of Informatics, University of Lugano, Switzerland
c.pautasso@ieee.org

Abstract We are currently developing a performance benchmark for Workflow Management System. As a first activity we are collecting real-world processes. However, to protect their competitive advantage, some companies are not willing to share their corporate assets. This work's objective is to propose a method ("BPELanon") for BPEL process anonymization in order to deal with the problem. The method transforms a process to preserve its original structure and runtime behavior, while completely anonymizing its business semantics. Anonymization is a complicated task that must meet the requirements we outline in this paper. Namely, we need to preserve the structural and executional information while anonymizing information such as namespaces, names (activity names, variable names, partner link names etc.), and XPath expressions that may reveal proprietary information. Furthermore, the names contained in the anonymized process should be chosen carefully in order to avoid conflicts, preserve privacy, and file-readability. Multiple dependency relations among process artifacts raise the challenge of fulfilling the aforementioned requirements, as a unique change in a file potentially leads to a flow of changes to other related process artifacts.

Keywords: Anonymization, BPEL, Workflows, Business Processes

1 Introduction

Given the fact that "process equals product" [3] most companies and business organizations are not willing to share their process models with academic researchers due to competitive reasons to protect their intellectual property. Since our first goal with the "BenchFlow" project[1] is to collect real-world business process models that can be later used to synthesize a Benchmark, we want to encourage sharing of models that are suitable for our purposes without revealing critical company information. The contributions of this work are as follows:

1. identify the requirements of anonymization methodology
2. propose a method ("BPELanon") that exports the anonymized process model containing the original BPEL process without its business semantics, but solely its executable structure

[1] http://www.iaas.uni-stuttgart.de/forschung/projects/benchflowE.php

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

2 Approaching the Problem

2.1 Requirements

The design of "BPELanon" must address the following initial list of requirements identified during our work in various research projects, and especially during our collaboration with industry partners: The main requirement and purpose of methodology is to:

R1: Support both pseudonimization and anonymization of data upon the user's choice. Pseudonimization is the technique of masking the data, while maintaining ways to the original data [1]. On the contrary, anonymization changes the critical data and makes it impossible to trace back the original version of data [4]. Providing the option of pseudonimization makes it possible for the originator to trace bugs or inconsistencies found in the anonymized file, and apply changes to the original process.

In order to satisfy [R1] a number of other requirements occur. These can be grouped to requirements that stem from the XML nature of BPEL:

R2: Scramble the company's sensitive information that can be revealed in activity names, variable names, partner link names, partnerlink type names, port type names, message names, operation names, role names, XSD Element names, namespaces, and XPath expressions. The name choice for these attributes is usually descriptive, and reflects the actual actions to which they correspond. So they can reveal a lot of the process semantics.
R3: The exported process model must not contain namespace information in incoming links to external web sites that reveal business information (backlinks)
R4: The exported process model must not contain names (including activity names, variable names, partner link names, partnerlink type names, message names, operation names, role names, and XSD Element names) with backlinks to business information
R5: The exported process model must not contain XPath expressions with backlinks to business information. If no custom XPath functions are used, [R5] is a consequence of requirement [R4].
R6: Remove description containers (comments and documentation) that reveal critical information and semantics.

BPEL-specific requirements:

R7: The exported process model must keep the structural information and executability
R8: The exported process must maintain an equivalent run-time behavior
R9: The exported process must maintain equivalent timing behavior

The following requirements are related to the renaming methodology that will be applied:

R10: It has to be ensured that the scrambled name prevents reverse engineering to get the original names. For example if data is encrypted with a known function (e.g. RSA, MD5) and we know the used key, then it is easy to obtain the original data.

R11: Names must be chosen in a way that conflicts are avoided between the original and exported file. For example an easy name choice would be to change each name with respect to its type followed by an ascending ID. For example the name of activity "Payment" could have been changed to the name "Activity1". Nevertheless, this way is not considered safe. "Activity1" could also have been a possible name choice for the original process model as it is a word frequently met in Business Process Management. This would lead to a sequence of conflicts. Which elements named "Activity1" correspond to the anonymized element and which to the one contained in the original process?

R12: The names must lead to an human-readable exported file. For example let's assume that we use UUIDs for name choice. That would lead to activity names such as: `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`. The exported file will not be easy to read for humans.

2.2 Challenges

This section analyzes the challenges that stem from the need to satisfy the requirements described in

2.3 Requirements

Each process specification is wrapped in a package which is a directory containing all deployment artifacts. At the minimum the directory should contain a deployment descriptor, and one or more process definitions (BPEL), WSDL, and XSD files[1]. Many dependency relations among files as shown in Fig. 1 increase the complexity of anonymization as small changes in a file may lead to numerous subsequent changes to other process artifacts [Challenge 1]. The complexity increased by the need to meet Requirement 2 [Challenge 2]. The renaming methodology also needs to be carefully examined in order to satisfy Requirements 9–12 [Challenge 3].

The BPEL-specific requirements reveal a new set of challenges that will be more complex to fulfill. How do data and data specific decisions affect the runtime behavior of the anonymized model? [Challenge 4]. How is BPEL lifecycle affected by anonymization? [Challenge 5]. To what extend will timing behavior be maintained? [Challenge 6]. These challenges will be addressed in future work.

Following the approach of "divide and conquer" the anonymization methodology followed for each artifact should be separately and carefully examined. In this paper we focus on the BPEL - WSDL anonymization aiming to satisfy [Challenges 1, 2, 3].

[1] `http://ode.apache.org/creating-a-process.html`

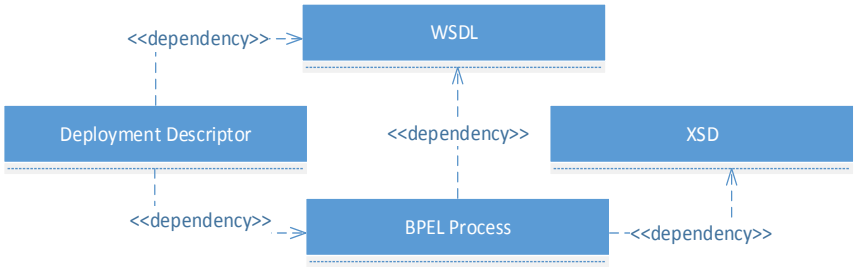


Figure 1. Dependency relations among artifacts to be anonymized

Fig. 2 shows a more detailed analysis of the occurring dependencies between the BPEL and WSDL artifacts. The grey entities represent the BPEL elements while the green entities represent WSDL elements. The directed associations that connect the members with each other show dependency between the entities. The arrow shows the "direction" of dependency. This means that the member to which the arrow leads is an artifact which creates high dependencies between the rest of the participating entities. Therefore when this member is changed the interconnected members should be accessed and changed correspondingly.

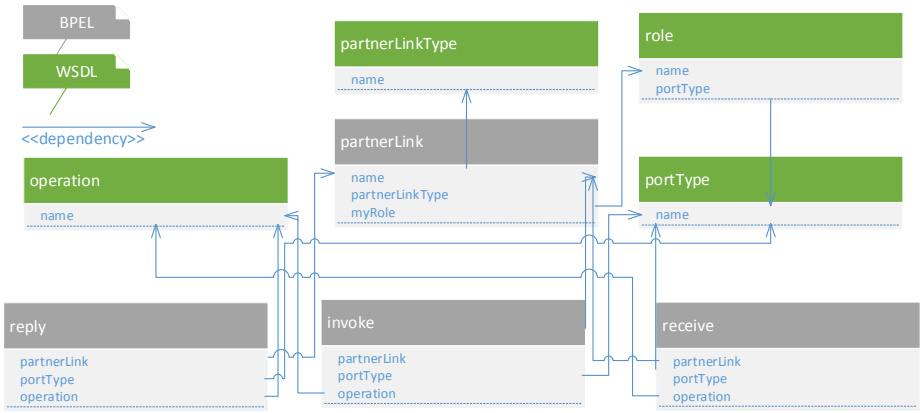


Figure 2. Dependencies between BPEL and WSDL files of a Business Process

3 Designing the Method

This section describes the methodology that is used for developing "BPELanon". Elements in a BPEL file can be divided into three groups:

– Free Elements Group: Elements that need to be anonymized, but are not bound to changes that occurred in other files.

- WSDL Bounded Group: Elements that need to be changed because they were bounded with elements that are changed in the WSDL file.
- Internally Bounded Group: Elements that need to be changed because they are bounded to other changed elements within the same file. Internally Bounded Groups can be found in both BPEL and WSDL files.

The anonymization of "Free Elements Group" is trivial. However, the anonymization of "WSDL Bounded Group" and "Internally Bounded Group" are more complex tasks. For its implementation we need a "Registry of Alterations". This is a registry of metadata that is created during the anonymization a file and logs the occurring changes. It must contain in the minimum the following information: the element's type, and the corresponding attributes' new and old data.

The main idea of the anonymization is to scan the documents (WSDL, BPEL does not matter) looking for element attributes that might contain semantics (critical attributes) and need to be scrambled, and adding them to the "Registry of Alterations" the old and new value. The information on which attributes are critical can be stored with metadata. Next we scan the documents looking for references to the scrambled elements and update their values. Below it is described the anonymization method for the "WSDL Bounded Group".

Anonymization starts with the creation of a metadata schema that reflects the interconnections shown in Fig. 2. Next we construct a "Table of References" that shows correlation of a BPEL process and its WSDL files. This is done by parsing the `<bpel:import>` annotations of the BPEL file. We then process the WSDL files, which contain the definitions for the artifacts that are referenced in BPEL. We run through each one of the WSDL files in "Table of References" and start anonymizing the attributes of the elements step by step. In order to fulfill [R8] the function of anonymization will pick random worlds of an English Dictionary [1]. As argued before a world of a well known human language will lead to more readable results than UUIDs. We only focus on the anonymization of critical attributes as not every attribute needs to be anonymized. By maintaining a "Registry of Alterations", we apply the subsequent changes to the BPEL. We have created an outer loop that repeats this process for each WSDL file. Another option would have been to parse all WSDL files and finally apply the changes to BPEL file in one parse. However WSDL files might have common names and this would lead to more complex solution. We have therefore chosen this safer although most likely more time consuming method.

At the end of the process "Table of References" and "Registry of Alterations" is destroyed if the tool is set to anonymize and not pseudonimize. The above procedure describes Algorithm 1. For the anonymization of the "Internally Bounded Group" a similar process needs to be followed.

4 Related Work

Attempts for anonymization can be found in various fields of computer science such as network security (filtering, replacement, reduction of accuracy etc. [6]) and

[1] `http://www.winedt.org/Dict/`

Algorithm 1 Anonymization process of BPEL-WSDL for "WSDL Bounded Group"

```
create TableOfReferences by parsing <bpel:import> annotations of BPEL
for all WSDL files W in tableOfReferences do
  for all elements E in W do
    a ← getCriticalAttributes(E)
    for all a do
      updateRegistryOfAlterations(E.type,a.type,a.data,"old")
      applyAnonymizationPattern(a.data)
      updateRegistryOfAlterations(E.type,a.type,a.data,"new")
    end for
  end for
  for all element E in BPEL file do
    a ← getCriticalAttributes(E)
    for all a do
      resultType ← findTypeOfInterconnection(E.type,a.type)
      a.data ← getNewValueOfAttribute(resultType,a.data) {from registryOfAl-
      terations}
    end for
  end for
end for
if anonymization then
  delete tableOfReferences
  delete registryOfAlterations
end if
```

database systems (data generation, encryption etc. [5], k-anonymity, l-diversity, and t-closeness[1]). These approaches cannot be applied to BPEL as they are tightly tailored to the architecture and principles of different technologies.

The tools XMLAnonymizer[2] and XMLAnonymizerBean[3] were found. XMLAnonymizer is a primary approach to anonymization that focuses on changing the attribute value of the XML file ([R4] partially covered). The XMLAnonymizerBean anonymizes elements and attributes by removing the namespaces of an XML file ([R3] partially covered). Overall, these utilities partially satisfy the requirements of "BPELanon". The "BPELanon" method is a more complex approach since it deals with all the requirements and challenges described in Sect. 2.

5 Conclusions and Future Work

In this paper we have proposed a method for the anonymization of BPEL processes. We focus on BPEL processes without extensions as experience shows

[1] http://arx.deidentifier.org/

[2] https://code.google.com/p/xmlanonymizer/

[3] http://help.sap.com/saphelp_nw04/helpdata/en/45/d169186a29570ae10000000a114a6b/content.htm

that BPEL is used widely in industry to implement workflows. There are more than 60 BPEL extensions available [2], but the processes we collected so far indicate that none of these extensions is used in real-world settings. We have analyzed a set of requirements and challenges that make process anonymization difficult. To address the requirements and challenges we suggest an algorithm that is a first approach to the methodology of business process anonymization. The main contribution of this paper is the design of a methodology with focus on BPEL anonymization.

In future work we will investigate what is the impact of anonymization to the BPEL process lifecycle, the ways that data and data dependent decisions are influenced by anonymization, and include timing behavior information into BPELanon methodology. The implementation of "BPELanon" has started, and will be tested with a set of workflows with various characteristics. The first release will be then distributed to companies for evaluation and usage. We intend to extend "BPELanon" in order to provide various options of anonymization, and anonymization valid for other languages. After collecting a sizable sample of anonymous process models, we will work on a method for "Statistical Analysis" that aims to calculate useful statistical information out of the BPEL process collection.

Acknowledgments This work is funded by the "BenchFlow" (LE 2275/7-1) project supported by German Research Foundation (DFG).

References

1. Federal Ministry of Justice: German Federal Data Protection Law (1990)
2. Kopp, O., Görlach, K., Karastoyanova, D., Leymann, F., Reiter, M., Schumm, D., Sonntag, M., Strauch, S., Unger, T., Wieland, M., Khalaf, R.: A Classification of BPEL Extensions. JSI 2(4), 2–28 (November 2011)
3. Leymannn, F.: Managing business processes via workflow technology. Tutorial at VLDB 2001 (11-14 September 2001)
4. Strauch, S., Breitenbücher, U., Kopp, O., Leymann, F., Unger, T.: Cloud Data Patterns for Confidentiality. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science, CLOSER 2012. pp. 387–394. SciTePress (2012)
5. Vinogradov, S., Pastsyak, A.: Evaluation of data anonymization tools. In: DBKDA 2012, The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications. pp. 163–168 (2012)
6. Yurcik, W., Woolam, C., Hellings, G., Khan, L., Thuraisingham, B.M.: Toward trusted sharing of network packet traces using anonymization: Single-field privacy/analysis tradeoffs. CoRR abs/0710.3979 (2007)

Where did I go wrong?

Explaining errors in business process models

Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
niels.lohmann@uni-rostock.de

Abstract. Business process modeling is still a challenging task — especially since more and more aspects are added to the models, such as data lifecycles, security constraints, or compliance rules. At the same time, formal methods allow for the detection of errors in the early modeling phase. Detected errors are usually explained with a path from the initial to the error state. These paths can grow unmanageably and make the understanding and fixing of errors very time consuming. This paper addresses this issue and proposes a more intelligible explanation of errors: Instead of listing the actions on the path to the error, only which decisions lead to it are reported and highlighted in the original model.

1 Introduction

Business process modeling is a sophisticated task and received a lot of attention in the past decades. With the advent of domain-specific languages and a growing scientific community, the act of creating and managing business process models has become a discipline on its own. Despite all efforts, design flaws may still occur. This can have different impacts, ranging from syntactically incorrect models, which are harder to understand, up to catastrophic faults and down times in the execution that yield to a loss of money or a legal aftermath. Consequently, a large branch of research focuses in the detection, correction, and avoidance of errors in business process models. Whereas plain control flow analysis is now well understood, other aspects such as data, business rules, or security may introduce more subtle flaws that are harder to detect.

Using the prominent *soundness* [1] property, we can classify existing approaches into three classes: (1) Some approaches exploit certain structural constraints of the business process model, for instance by focussing on workflow graphs that only consist of AND/XOR-gateways, for instance [10]. (2) Other approaches rely on the definition of soundness which can be defined in terms of standard Petri net properties such as boundedness, liveness, or the existence of place invariants [11]. The two mentioned approaches are *domain-specific* in the sense that they exploit the fact that they investigate business process models. In contrast, (3) general purpose verification tools (usually called *model checkers* [3]) can check all kinds of properties as long as they can be expressed in terms of temporal logics. As this is the case for soundness, these tools are also applicable for the verification of business process models.

Due to the ongoing evolution of business process modeling languages, the growing number of aspects that need to be covered by a business process model,

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

or the trend toward executable business process models, the verification of business process models has become a moving target. As a consequence, specific approaches may become inapplicable for novel demands, leaving only general purpose approaches as stable tools for the future.

Problem description. In principle, a model checker takes a formal model (e.g., a Petri net) and a formal description of the property to check (usually described by temporal logic formula φ) as input and tries to prove the property by an exhaustive investigation of the model's states. In case the property is violated (e.g., a deadlocking state is detected), a path π to this error state is reported [3]. The path contains all actions of the model that need to be executed to reach the error state from the initial state. Due to this operational nature of paths, the scenario that led to the error can be simulated. It is furthermore possible to explain the scenario in terms of the original model; that is, to map the states of the Petri net back to events of a BPMN model.

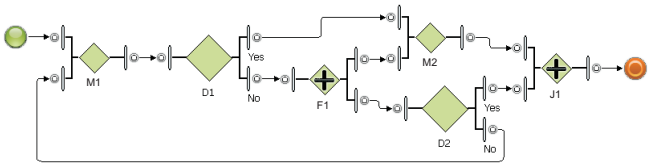
Unfortunately, the size of the paths correlates with the size of the model and paths of industrial models can thus be very long and hardly understandable. Furthermore, the path can contain a lot of irrelevant or diverting information that makes the comprehension of the error very difficult. For instance, the path usually contains actions that only "set up" the process (e.g., initializations and login procedures). These inevitable actions are certainly *necessary* to be able to reach the error state, but are usually not the *cause* of it. Another aspect that makes paths hard to understand is the fact that business process models may span several components where activities are executed in parallel. On the path, these originally unordered activities are reported in a fixed — and possibly arbitrary — order which may yield confusion due to unintuitive error descriptions.

Contribution. This paper addresses the mentioned problems by shortening paths by focussing on the choices made rather than on each individual action. We shall use a large case study as experimental evaluation of our proposed approach.

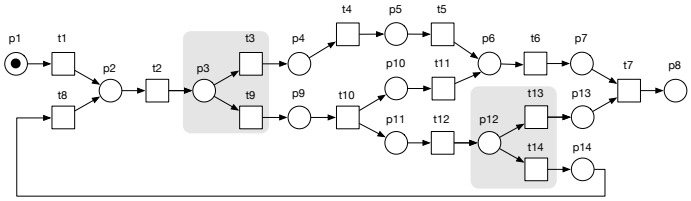
2 Model checking Petri nets

Business process modeling languages are usually semiformal and hence are not directly applicable to a mathematically rigorous proof of correctness criteria. However, the operational semantics can be captured in formalisms such as Petri nets or process calculi. With the advent of executable languages such as WS-BPEL 2.0 or BPMN 2.0, such a formalization became much easier, because a precise execution semantics yielded more careful language specifications. In fact, for most of today's languages from industry or academia, translations into *Petri nets* [9] exists [7].

Example. Figure 1(a) depicts a small business process model from [4] which contains two subtle control flow errors: a lack of synchronization and a local deadlock. Its translation into a Petri net is shown in Fig. 1(b). As we see, the Petri net's structure is very similar to the original model.



(a) business process model from [4]



(b) Petri net model

Fig. 1. A business process model (a) and its translation into a Petri net (b)

Model checking [3] is an approach to prove that a system satisfied a given correctness criterion; for instance soundness, the absence of a deadlocking state, the presence of a sound process configuration, correct data life cycles, or compliance to business rules. In contrast to *theorem provers*, which sometimes need manual inputs, or *testing*, which can only prove the existence of errors, but never their absence, model checking is an automated and complete way to investigate systems.

For the remainder of the paper, we use model checking tool LoLA [12] that takes a Petri net N and a temporal logical formula φ as input. If the formula is satisfied by the Petri net (e.g., if the Petri net is sound), this is reported as "yes" to the modeler. In case the formula is violated (e.g., a deadlocking marking m is found), this is reported as "no" to the modeler. In addition, a path $\pi = t_1 \cdots t_n$ is given to the modeler which explains how m is reachable from the initial marking m_0; that is, $m_0 \xrightarrow{t_1} \cdots \xrightarrow{t_n} m$. Depending on the nature of the formula φ, the marking reached by the reported path either is a proof that the formula is not satisfied by the behavior of the Petri net N and is called a *counterexample* or marking itself is the proof that the formula is satisfied (e.g., if φ expresses the reachability of that marking m) and is called a *witness*. In this paper, we do not distinguish the semantics of the marking m and always refer to m as *goal marking*.

Example (cont.). The business process from Fig. 1(a) has a lack of synchronization. This can be detected by checking the Petri net from Fig. 1(b). The following path π describes how a marking m can be reached which puts two tokens on place p_6.

$$\pi = t_1\ t_2\ t_9\ t_{10}\ t_{11}\ t_{12}\ t_{14}\ t_8\ t_2\ t_3\ t_4\ t_5 \qquad m = \{p_6 \mapsto 2\}$$

The path contains 12 transitions. In the remainder of this paper, we use this path to exemplify the proposed reductions.

It is worthwhile to mention that model checking suffers a devastating worst case complexity due to the well-known state explosion problem which yields reachability graphs with exponential blow-ups compared to the size of the models. However, even industrial business process models can be model checked in few microseconds, because heuristics that fight the state space explosion proved to be very effective in this domain [4].

3 Representing paths by made choices

3.1 The problem: long paths = big problems

In the remainder of the paper, we focus on the following problem:

> Given a path π to a goal marking m of a Petri net model N, how can the reason for the error modeled by m be briefly and comprehensively explained to the modeler of N?

Apparently, π describes how the goal marking m can be reached from the initial marking m_0 of N. Consequently, reporting the transitions of π together with the intermediated markings to the modeler should help to understand the reasons m was reached. Unfortunately, this approach is futile in case π contains dozens of transitions. The reasons for such long paths are:

Detours: Model checkers usually investigate the markings of a Petri net in a depth first search[1]. As a result, the reported paths do not need to be optimal and may contain some transitions that model "detours" in the reachability graph that do not contribute in the actual reaching of the goal marking.

Interleaving of concurrent transitions: A marking of N may activate two transitions t_1 and t_2 which are not mutually exclusive. That is, firing either transition first does not disable the other one. A typical reason for this is that t_1 and t_2 do not share any resources. Consequently, the order in which t_1 and t_2 occur on the path π is arbitrary. If each transition belongs to different components of the underlying business process model, then these arbitrary interleaving of the transitions may be irritating to the modeler if she tries to understand the path π. In the example path, transition t_{11} and t_{12} are concurrent and the reported order in path π (t_{11} before t_{12}) is arbitrary.

Indisputable parts: Though the path π is an actual proof *that* the goal marking m can be reached in N, not every transition on the path is an actual cause of m. In the example process, *any* path will begin with firing t_1 and hence does not need to be reported to the modeler as reason for an error.

[1] Breadth-first approaches are not applicable to many classes of formulae.

3.2 The solution: don't report the obvious

To tackle the problem of long paths with redundant or unhelpful information, we shall exploit two aspects to shorten paths in the remainder of this section: *progress* and *conflicts*.

Progress is the assumption that the model never "gets stuck" in case a transition is activated. That is, if a marking activates one or more transitions, then this marking is eventually left by firing on of these transitions. Progress is a natural assumption for business process models in which the execution of tasks also cannot be postponed indefinitely. Though the actual occurrence of message or timer events cannot be precisely predicted, the respective states are always assumed to be eventually left by the modeled actions.

A *conflict* is a situation in which there exist more than one possible continuations (e.g., an XOR gateway). In terms of Petri nets, it is a marking in which two transitions t_1 and t_2 are enabled, but after firing either of them, the other transition is disabled. This situation is dual to concurrent transitions (see above) that do not disable each other. A detailed discussion of these aspects can be found in [9].

The combination of these aspects brings us to the following intuitive observation: *Only the conflicts on the path π carry information on how to reach the goal markings.* Any other marking m on the path between the initial and the goal marking either (1) enables no transition: Then this must be the goal marking itself, because it has no successor marking. Alternatively, (2) marking m enables exactly one transition: Then this transition is eventually fired due to the assumption of progress. Consequently, this transition does not need to be reported to the modeler as its firing was already determined by the previous transition on π thad lead to m. Finally, (3) marking m enables several concurrent transitions. These transitions may fire independently, and if all of them are on π, then the exact order is arbitrary.

Example (cont.). The conflicts of our running example are shaded gray in Fig. 1(b): transitions t_3 and t_9, as well as t_{13} and t_{14} are conflicting. As a result, we can reduce the path π as follows:

$$\pi_{\text{reduced}} = t_9\ t_{14}\ t_3 \qquad m = \{p_6 \mapsto 2\}$$

The firing of all other transitions is clear from the context from the intermediate markings and the assumption of progress. Note that the transition names need to be translated back into the terms of the original model. A different representation of π_{reduced} could be: "After (1) decision D1: *No*, (2) decision D2: *No*, and (3) decision D1: *Yes*, a lack of synchronization occurs after after merge M2." This is depicted in Fig. 2.

3.3 Experimental results

To evaluate the path reduction algorithm, we applied it to a large collection of industrial process models created by IBM customers using the *IBM WebSphere*

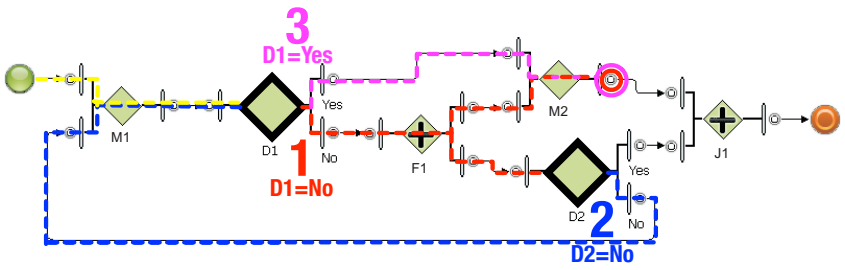


Fig. 2. Mapping back the reduced path run to the original process model

Business Modeler. The models were first presented in a report by Fahland et al. [4], where the 1386 process models were checked for soundness using different approaches. As a general-purpose model checker, *LoLA* [12], took part in this investigation, the process models were also translated into Petri nets.[2] The models are partitioned into five libraries (A, B1, B2, B3, C) and stem from different business areas, ranging from financial services, automotive, telecommunications, construction, supply chain, health care, and customer relationship management.

As experiment, we replayed verification checks reported in [4,2] and reduced the generated paths. The results summarized in Table 1–3 report are promising: we report reductions between 77 % and 95 %, leaving average reduced path lengths between 2 and 7 transitions. Though the reduced paths consist of Petri net transitions, it can be easily translated back into the nomenclature of the original model as demonstrated in Fig. 2.

4 Further reduction: remove spurious conflicts

In the previous section, we showed how paths to errors in business process models can be reduced by only reporting conflict transitions. This reduction decided, for each marking that activates a transition, whether conflicting transitions are also activated. This check is local in the sense that it is not checked whether those transitions that were not taken in the decisions actually could have avoided the next conflict transition on the path.

Intuitively, a transition t_i on a reduced path π is a spurious conflict iff every transition t in conflict to t_i eventually reaches the marking m_{i+1} which enables the next transition t_{i+1} on path π. In this case, choosing any transition from the ith conflict will eventually enable the next conflict on the path to the goal state. Consequently, reporting the spurious conflict t_i is of little help to the modeler to understand the error itself.

The check for spurious transitions defined above can be straightforwardly be implemented using a model checker.[3] We integrated this check as postprocessing

[2] The original models and their Petri net translations are available for download at `http://service-technology.org/soundness`.

[3] We check whether N with initial marking m_i' satisfies the CTL formula $\varphi = \mathbf{AF}\, m_{i+1}$.

Table 1. Paths from the checks for local deadlocks [4]

library	A	B1	B2	B3	C
avg. path length before / after	17.51 / 1.83	17.52 / 2.11	16.06 / 1.54	20.34 / 1.67	13.40 / 2.30
max. path length before / after	53 / 8	66 / 7	56 / 6	54 / 5	21 / 3
sum of path lengths before / after	1699 / 178	1419 / 171	1349 / 129	1688 / 139	134 / 23
reduction	89.52 %	87.95 %	90.44 %	91.77 %	82.84 %

Table 2. Paths from the checks for lack of synchronization [4]

library	A	B1	B2	B3	C
avg. path length before / after	30.83 / 3.17	10.47 / 0.66	12.16 / 0.68	11.50 / 0.59	51.00 / 7.57
max. path length before / after	89 / 13	52 / 7	100 / 8	103 / 14	120 / 17
sum of path lengths before / after	1079 / 111	1047 / 66	1459 / 82	1507 / 77	357 / 53
reduction	89.71 %	93.70 %	94.38 %	94.89 %	85.15 %

Table 3. Paths from the checks for noninterference [2]

library	A	B1	B2	B3	C
avg. path length before / after	12.06 / 2.79	13.82 / 2.55	18.13 / 2.33	14.27 / 2.55	11.27 / 2.33
max. path length before / after	44 / 7	70 / 7	95 / 7	95 / 7	27 / 3
sum of path lengths before / after	19699 / 4557	5707 / 1054	13835 / 1777	17494 / 3130	169 / 35
reduction	76.87 %	81.53 %	87.16 %	82.11 %	79.29 %

step after reducing the paths as described in the previous section. Note that executing a model checker can be very time and memory consuming. However, even if a check is not finished with a reasonable amount of resources, we just failed to proof whether a conflict is spurious and can continue with the investigation of the next transition. That said, the postpocessing can be aborted at any time — any intermediate result is still correct.

We applied the reduction of spurious conflicts to the case studies described in the previous section. Table 4–6 summarize the results. In all three experiments, the paths could be further reduced by 50–86%. Note that in some cases, the check for spurious conflicts has been aborted after more than 2 GB of memory were consumed. In these cases, the conflict was kept in the path and the check proceeded with the next conflict.

5 Concluding remarks

Related work. The analysis and verification of business process models is a broad field of research. Consequently, there exists a variety of domain-specific approaches (e.g., the decomposition of workflow graphs into SESE regions to check soundness [10]). However, we are not aware of other approaches that postprocess error information from general purpose model checkers to explain these errors to the modelers.

Related to the presentation of error information is the automated correction of flawed business process models [6,5]. These approaches use similarity metrics

Table 4. Reduced paths from the checks for local deadlocks

library	A	B1	B2	B3	C
avg. path length before / after	1.84 / 0.91	2.11 / 0.67	1.54 / 0.57	1.67 / 0.41	2.30 / 0.90
max. path length before / after	8 / 2	7 / 1	6 / 1	5 / 1	3 / 1
sum of path lengths before / after	178 / 88	171 / 54	129 / 49	139 / 34	23 / 10
reduction	50.56 %	68.42 %	62.79 %	75.54 %	60.87 %
aborted checks	1	0	0	0	0

Table 5. Reduced paths from the checks for lack of synchronization

library	A	B1	B2	B3	C
avg. path length before / after	3.17 / 0.86	0.66 / 0.17	0.68 / 0.14	0.59 / 0.09	7.57 / 1.00
max. path length before / after	13 / 2	7 / 2	8 / 2	14 / 2	17 / 2
sum of path lengths before / after	111 / 30	66 / 17	82 / 17	72 / 12	53 / 7
reduction	72.97 %	54.55 %	79.27 %	84.42 %	86.79 %
aborted checks	1	4	0	0	4

Table 6. Reduced paths from the checks for noninterference

library	A	B1	B2	B3	C
avg. path length before / after	2.79 / 0.99	2.55 / 0.75	2.33 / 0.55	2.55 / 0.63	2.33 / 0.40
max. path length before / after	7 / 2	7 / 2	7 / 2	7 / 2	3 / 1
sum of path lengths before / after	4557 / 1614	1054 / 310	1777 / 423	3130 / 772	35 / 6
reduction	64.58 %	70.59 %	76.20 %	75.34 %	82.86 %
aborted checks	12	4	4	7	0

to find a correct business process model which maximally resembles the flawed model. These approaches have the benefit of avoiding lengthy manual correction steps altogether.

Future work. In this paper, we focused on reducing paths to error states and neglected the retranslation into the original business process model. Visualizations such as Fig. 2, possibly enriched with animations, need to be automated and evaluated by business process modelers. Here, understandability criteria [8] could be of great value. However, this was out of scope of this paper which aimed at evaluating the idea of using conflicts to reduce paths with three experimental setups checking different correctness criteria with thousands of industrial business process models.

We see in this paper a first step toward a diagnosis framework which uses general purpose verification tools to verify business process models. As motivated in the introduction, domain-specific approaches are very closely coupled to the structure or the property under investigation, but may become inapplicable for future developments. In contrast, the modularization (a translation into Petri nets as frontend, a general purpose model checking tool as middleware, and a diagnosis framework as backend) may be more flexible when it comes to novel business process languages and properties.

References

1. W. M. P. v. d. Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
2. R. Accorsi and A. Lehmann. Automatic information flow analysis of business process models. In *BPM 2012*, LNCS 7481, pages 172–187. Springer, 2012.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
4. D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Instantaneous soundness checking of industrial business process models. In *BPM 2009*, LNCS 5701, pages 278–293. Springer, 2009.
5. M. Gambini, M. La Rosa, S. Migliorini, and A. ter Hofstede. Automated error correction of business process models. In *BPM 2011*, LNCS 6896, pages 148–165. Springer, 2011.
6. N. Lohmann. Correcting deadlocking service choreographies using a simulation-based graph edit distance. In *BPM 2008*, LNCS 5240, pages 132–147. Springer, 2008.
7. N. Lohmann, H. Verbeek, and R. M. Dijkman. Petri net transformations for business processes – a survey. *LNCS ToPNoC*, II(5460):46–63, 2009.
8. J. Mendling, H. A. Reijers, and J. Cardoso. What makes process models understandable? In *BPM 2007*, LNCS 4714, pages 48–63. Springer, 2007.
9. W. Reisig. *Petri Nets*. Springer, EATCS Monographs on Theoretical Computer Science edition, 1985.
10. J. Vanhatalo, H. Völzer, and F. Leymann. Faster and more focused control-flow analysis for business process models through SESE decomposition. In *ICSOC 2007*, LNCS 4749, pages 43–55. Springer, 2007.
11. H. M. W. Verbeek, T. Basten, and W. M. P. v. d. Aalst. Diagnosing workflow processes using Woflan. *Comput. J.*, 44(4):246–279, 2001.
12. K. Wolf. Generating Petri net state spaces. In *ICATPN 2007*, LNCS 4546, pages 29–42. Springer, 2007.

Message Assertions and Predicate-Based Control-Flow Unfolding Revisited

Thomas S. Heinze[1], Wolfram Amme[1], and Simon Moser[2]

[1] Friedrich Schiller University of Jena, Institute of Computer Science
[t.heinze,wolfram.amme]@uni-jena.de
[2] IBM Research & Development Böblingen
smoser@de.ibm.com

Abstract. In our previous work, we considered, on the one hand, the predicate-based unfolding of a business process's conditional control flow and, on the other hand, the introduction of assertions for the contents of messages exchanged between processes. In this paper, we will sketch how both approaches can be smoothly combined for the automated provisioning of precise low-level Petri net models of business processes.

1 Introduction

The quality of Petri-net-based verification of business processes is tightly coupled to the precision of the process model used. In particular, if verification targets the soundness or controllability of full-specified processes, i.e., executable BPMN or WS-BPEL processes, the Petri net model of a process must reflect the process's control flow as well as process data relevant to the control flow. Otherwise, a thus imprecise process model comprises the danger of an erroneous verification. Yet, a precise and verifiable Petri net model can not be given in the general case due to the Turing-completeness of languages BPMN and WS-BPEL.

In our previous work [2,3], we presented methods which aim at the generation of precise low-level Petri net models for business processes. At the core of the methods is the semantic-preserving transformation of a process's data-based choices into unconditional control flow, such that process data relevant to the choices' conditions, and therefore verification, does not need to be included in the Petri net model. Despite not being effectful in all cases, the methods allow for the generation of precise process models in a number of substantial cases.

In this paper, in order to widen the range of cases our methods are successfully applicable, we integrate the individual methods, namely predicate-based control-flow unfolding [3] and assertions for the contents of incoming messages [2], into a consolidated technique. Specifically, we propose:

- the use of predicate-based abstractions on process data to derive assertions,
- the generalized application of assertions to incoming and outgoing messages,
- the definition of a data-sensitive communication model, which fits with existing algorithms and theories for Petri-net-based process verification.

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

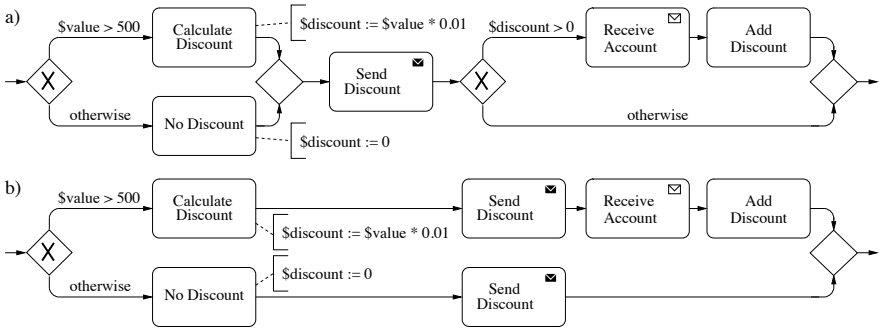


Fig. 1. Snippet of distributed business process (a) and its unfolding (b)

2 Predicate-Based Control-Flow Unfolding

In principle, predicate-based control-flow unfolding [3] aims at resolving the data-based choices of a business process. To this end, a predicate-based abstraction for process data is derived by static analysis, which is afterwards used to evaluate and eliminate the choices' conditions. However, a single choice's condition can be evaluated to true on one path of the control flow and to false on another, according to the values of process data valid at the respective path. Therefore, the process's control flow is unfolded in a controlled fashion, such that differing (abstract) values of process data are assigned separate control flow paths.

Figure 1 a) gives an example: It shows a snippet of a business process, in which the discount for a customer's order is calculated (`$discount`), sent to the customer, and booked to the customer's account if greater than zero. In Figure 1 b), the result of unfolding the snippet is shown. While unfolding, abstractions `$discount` > 5 and `$discount` $= 0$ were automatically derived for `$discount`'s value using static analysis and assigned individual control flow paths. Thereupon, the data-based choice with condition `$discount` > 0 could be successfully evaluated and resolved based on the abstract values.

3 Data-Sensitive Message Channels

However, mapping the unfolded snippet in Figure 1 b) to a Petri net, using the Petri net semantics of [6], still does not result in a precise process model. In the snippet, an order with a value greater 500 requires the customer to send another message while an order smaller or equal 500 implies no further interaction. Unfortunately, this relation is not reflected in the Petri net, since the snippet's (remaining) choice is therein modeled by nondeterminism, so that a partner is not able to determine if it is expected to send another message or not.

In [2], we introduced assertions for incoming messages, based upon simple relational expressions over constants, in order to distinguish between messages with different message contents. Apparently, this approach can as well be applied to outgoing messages and arbitrary predicates. Thus, a (data-sensitive) message channel C_P is now assigned an assertion $\forall x\colon P(x)$, where x denotes the message

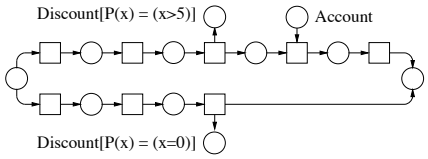


Fig. 2. Precise Petri net model, i.e., open workflow net, for snippet in Figure 1 b)

content and P a first-order predicate. As a remark, this definition can be easily extended to messages containing multiple data items and to assertions with formulas over more than one predicate. Doing so in particular allows for reusing the predicate-based abstraction for process data, which has been derived and utilized in the process of unfolding, as assertions for (outgoing) messages.

For the example in Figure 1 b), message channels with assertions $\forall x\colon (x > 5)$ and $\forall x\colon (x = 0)$ are introduced for outgoing message `Discount`. Consequently, mapping the snippet to a Petri net now yields a precise process model (Figure 2), such that a partner is enabled to determine whether it is expected to send another message or not by considering the assertions assigned to message `Discount`.

Since existing Petri net models for (distributed) business processes do not support assertions for message contents, we need to extend their communication model. For open workflow nets [4], such an extension is obtained by modifying the composition operator so that it considers the assertions while gluing interface places. Note that this can be challenging, i.e., requires the use of a SMT solver, for messages with differing assertions. Further, each incoming message is attached an additional channel with assertion *true*, conflicting the message's other data-sensitive channels, and conventional channels are assigned assertion *true* too.

4 Related Work

Existing approaches for mapping business processes to Petri nets either omit process data entirely or restrict themselves to finite data [1,4,5]. This also applies to the process-to-Petri-net compiler in [4], where, despite being defined on high-level nets, a low-level net is generated in which data-based choices are mapped to nondeterminism and messages are modeled as indistinguishable tokens. High-level nets support data modeling, though, can in general not be verified in the presence of infinite data [7]. To the authors' knowledge, there is no other work done on the application of predicate abstraction to a process-to-Petri-net-mapping.

5 Conclusion

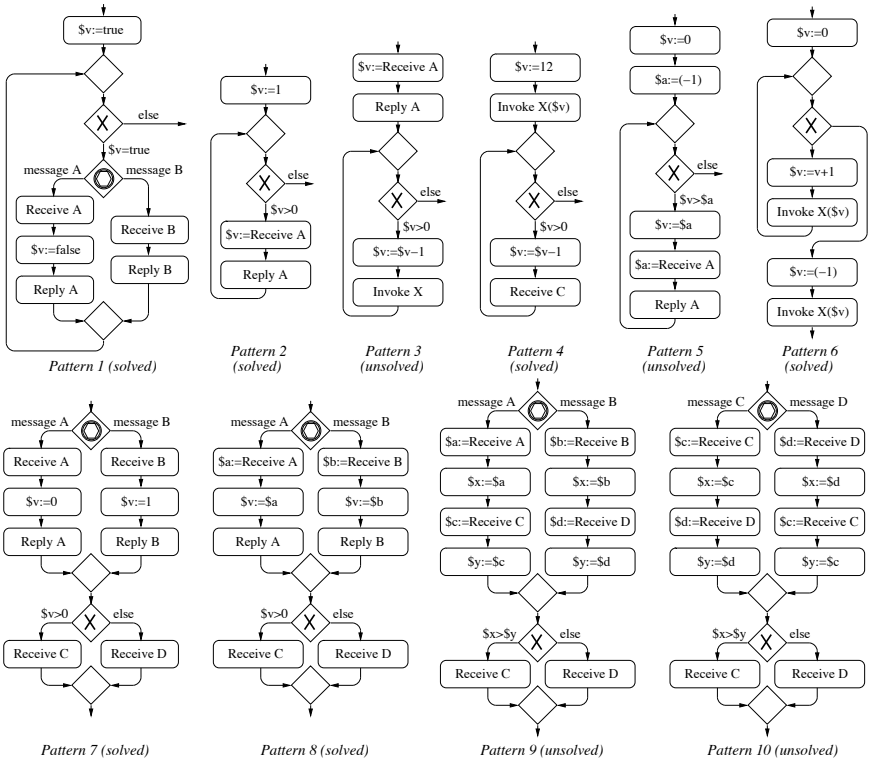
In this paper, we have sketched the smooth integration of our previously introduced methods into a consolidated technique by way of an example. Using the technique then allows for generating more precise Petri-net-based process models for business processes in a significant number of cases. In particular, applying the technique to a set of problematic patterns of distributed processes, supplied by an industrial partner (see Appendix A), revealed the successful generation of

precise process models for six out of ten process patterns. However, the thorough assessment of efficacy remains subject to future work. To this end, we plan to evaluate the consolidated technique using our process-to-Petri-net compiler.

References

1. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology 50(12), 1281–1294 (2008)
2. Heinze, T.S., Amme, W., Moser, S.: Process Restructuring in the Presence of Message-Dependent Variables. In: ICSOC Workshops. pp. 121–132. Springer (2011)
3. Heinze, T.S., Amme, W., Moser, S.: Control Flow Unfolding of Workflow Graphs Using Predicate Analysis and SMT Solving. In: ZEUS 2013. pp. 1–8 (2013)
4. Lohmann, N.: A Feature-Complete Petri Net Semantics for WS-BPEL 2.0. In: WS-FM 2007. pp. 77–91. Springer (2008)
5. Ouyang, C., Verbeek, E., van der Aalst, W.M.P., Breutel, S., Dumas, M., ter Hofstede, A.H.M.: Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming 67(2–3), 162–198 (2007)
6. van der Aalst, W.M.P., Hirnschall, A., Verbeek, H.M.W.: An Alternative Way to Analyze Workflow Graphs. In: CAiSE 2002. pp. 535–552. Springer (2002)
7. van der Aalst, W.M.P., Stahl, C., Westergaard, M.: Strategies for Modeling Complex Processes using Colored Petri Nets. In: ToPNoC VII, pp. 6–55. Springer (2013)

Appendix A – Process Patterns



Support of Decision Tasks in Business Process Improvement

Ekaterina Bazhenova

Hasso Plattner Institute at the University of Potsdam
ekaterina.bazhenova@hpi.uni-potsdam.de

Abstract. The vigorous technological development of the world is a key driver of the continuous improvement of business processes at the enterprise of today. For staying competitive, the companies have to be able to adapt to changes of business environment quickly and effectively. The conventional approaches to business process reengineering are based mostly on activity flows, and mostly not considering data of business processes. In the paper we present a hybrid approach to business process improvement, in which both activities and data of business processes are taken into account. As well, we introduce a concept of the decision task as a special kind of the business process and propose an approach for its improvement based on decision theory, which grasps all the stages of the business process lifecycle.

1 Motivation

Business process management has proved itself as a sustainable management approach and a competitive advantage of the enterprises where it is used. However, such factors as rapidly emerging technologies and instability of the market-driven economies, push enterprises to continuously rethink the ways of running their businesses. For facing such challenges, the industry needs efficient methodologies of business process improvement, which will provide the companies the ways to effectively and quickly apply the needed changes.

The interest to business process reengineering is reflected notably in a large variety of scientific literature where the fundamental approaches to business process redesign are presented [5,1]. The literature analysis shows that the most existing improvement techniques are bound with the activity flows and mostly do not take the business process data into account. However, "process and data are equally important for business process management" [2], and taking the business data into consideration will complement the approaches to business process improvement.

The pursuit of ways of effective business process transformation shows that similar business processes often have different execution outcomes as a result of uncertainties of the business environment. For example, the business process of time management at the enterprise can change to a great extent depending on the information about the preferences of the customer for the scheduling of meetings. In reality, making a choice under uncertainty can result in unnecessary time

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

and money investments and non-optimal business process management, and the companies need effective ways of dealing with such challenges. This has become a premise for us to apply the decision theory for business process management as a mechanism for dealing with processes of making decisions under uncertainty.

More specifically, we present the generic scheme of business process reengineering and we introduce the decision tasks as a special kind of business processes by mapping the objects of the decision theory and the business processes. Furthermore, we provide the approach of improvement of the decision tasks at all the stages of the business process lifecycle, by defining the payoff function of the process and proposing the outlook for its optimization.

2 Business Process Improvement

The business process lifecycle traditionally starts with the design and analysis phase, during which the business processes are identified and reviewed, mostly at the modeling level [6]. In Figure 1 we present our approach to business process improvement, which addresses this phase.

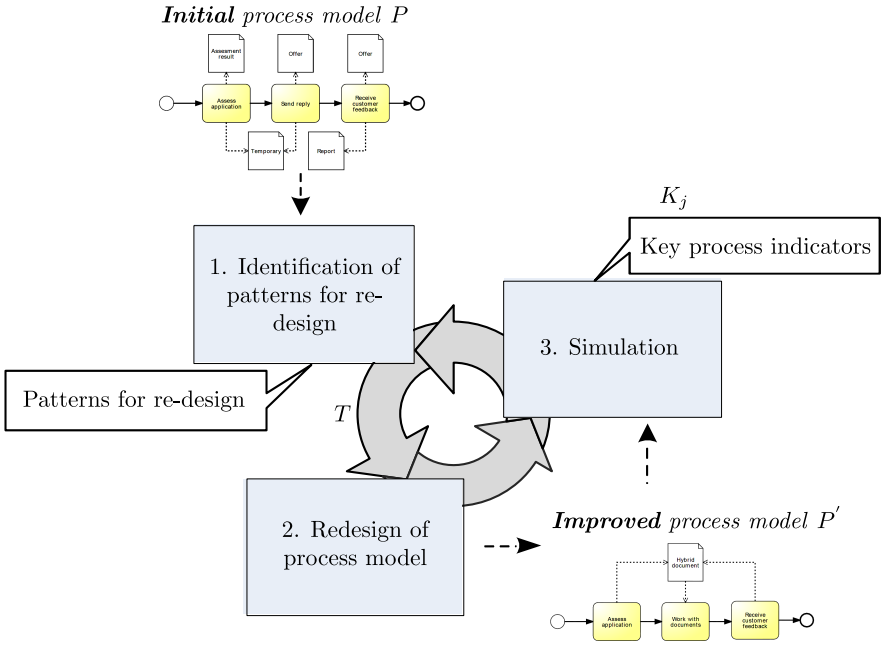


Fig. 1. Generic scheme of the business process improvement

As one can see from the figure, the object being exposed to redesign is a process model. It can be chosen, for example, by the business analyst which

supposes that the current process model is not efficient. The first step of the improvement approach is reviewing if the initial process model P contains a pattern for redesign, which could be be potentially improved by a transformation T, yielding as an outcome an improved process model P'. The logic and structure of the pattern and the transformation approach are provided in such a way that it refines the business process, which we will present in more details later in this section. For testing if the improved process model P' is more efficient than the initial process model P, the simulation phase is needed. For that, the key process indicators K_j, such as costs and times, should be introduced. The approach to simulation of the redesigned process model is planned for future work and the details of it are not discussed in the current paper. The whole improvement process can be repeated in case the changes are needed again, which is reflected in Figure 1 by the continuous loop sign.

Another notion which should be considered at all phases of the business process lifecycle, is the data which is inseparably associated with the business processes. Conventionally, the data in the enterprises is being stored in the corporate databases or data warehouses of ERP-systems [4]. In such approaches the optimization of the data usage is being reached by migration of data from different sources of the enterprise into a single storage which comprises of several stages such as data extraction, cleansing, transforming, indexing and loading. This migration is a technical processing of data based on the set of properties that guarantee that database transactions are processed reliably. However, such processing does not take into account the business context of the processes which are bound with the data objects. In our approach we are looking at the contextual dependencies of data objects within the business process and the ways to optimize them.

3 Decision Tasks

As it was already mentioned in the introduction to the paper, many business processes face the uncertainties of the business environment and the decision theory is a tool which is focused on dealing with such challenges. For instance, in the business process of scheduling the time of the meeting the participants might not know the preferences of the others so that they have to make the choice under uncertainty which can result in the longer time of the decision process.

For the sake of approaching the aforementioned challenge, the particular kind of the business processes, which incorporate decision making, should be introduced. With regard to the foundations of decision theory [3], we present here some of the notions which are relevant for business process management:

1. The core setting of the decision theory is an occurrence of a subject or a *decision maker* whose goal is to make an optimal choice between the *set of alternatives*. For example, in the business process scenario of scheduling the meeting, the participants of the process choose between several alternative dates, proposed by the organizer.

2. One of the main assumptions of the decision theory is that any realization of the alternatives resulting from a decision can be *compared*. Another important assumption is that the decision maker is making choice in a *rational way*. Both these assumptions are valid for business process management as its principle is to avoid ambiguities at the modeling level and ensuing consequences at the execution level which might result in non-optimal usage of resources.

The notions presented above became a premise for setting the special kind of business process models to which we refer as the *decision tasks*. The generic structure of the decision task is shown in Figure 2. Here the set of alternatives is

Fig. 2. Structure of the decision task

presented as the collection input data object and the final decision is presented as the collection output data object. An important assumption with the regard to the decision tasks is that the data presented by the output data object "Decision" is a subset of the input data object "Alternatives". More specifically, that could be represented as following: the values of data attributes of the output data object are subset of the data attributes of the values of the input data object. Whether this subset of the data object "Decision" consists of one or several data attributes or it contains only an empty element, depends on the context of the business process. For example, the owner of the business process of scheduling of meetings at the enterprise can define whether it is acceptable to schedule several dates of the series of meetings or exactly one date of a meeting should be identified. Another assumption for the introduction of the decision tasks is that the decision makers exist and they are represented in the process model by the resources bound to the decision task.

Now that the definition of the decision task was introduced, the approach for improvement of such a process model can be suggested. Below we present the approach for business process improvement which consists of three consequent phases corresponding to the stages of the aforementioned scheme (see Figure 1):

1.1 Analysis of Business Process Model.

The business process improvement scheme can be launched when the business analyst of the enterprise decides that the current business process is not efficient according to some indicators.

Example. Organization of the scheduling of meetings at the enterprise is being done by a secretary which writes a personal e-mail to every participant of the meeting, collects the responses, chooses the date and sends it back to participants for confirmation. If less than required minimum of people confirm their participation, the process repeats. The structure of the process model

is non-optimal and its execution produces many data artifacts (e-mails).The business analyst is aware about existence of the scheduling software and he wants to explore if the business process model can be redesigned.

1.2 Detection of Decision Task. As the next step in the first stage in the business process improvement scheme, the identification of the pattern for redesign is needed. More specifically, it should be identified if the current process model P represents a decision task with the structure presented in Figure 2.

Example. The business process of organizing the scheduling of meetings at the enterprise represents the decision task. Here the set of alternatives is the set of dates to be chosen and the participants are decision makers which choose the final date. If their preferences are collected by the secretary, they do not know about the choices of each other, so they make their choice under uncertainty.

2.1 Definition of Payoff Function. The improvement of the internal structure of "Business Activities" of the decision task can be done by the application of the decision theory methods. That could be done by investigating the data bound to the decision task, more specifically, the attributes of data objects. The persons or other resources, involved into the execution of the decision task, can be viewed as decision makers. And, according to the assumption of rational behavior of decision makers, their goal is to maximize the *expected payoff* of the decision task. Therefore, the assigned goal of this stage is to set the payoff function of the decision task.

Example. The payoff function in the business process of scheduling of meeting could be the time saved by participants to agree on the final date of the meeting.

2.2 Optimization of Decision Task. In such a way, we reduced the challenge of business process improvement to the task of maximization of the expected payoff of the decision task. The approach for solving this challenge, or, more specifically, the transformation T, could be done, for example, by granting access for the decision makers to the data produced by each participant. Then the decision makers will be able to estimate the payoff of their choices more precisely which might result in reducing the time spent on the decision making process and, consequently, in the time spent on the execution of the whole scheduling process. The outcome of the transformation T of the initial process model P is the improved process model P'.

Example. In case of choosing the date for the meeting, a dedicated web-page containing the table with all the alternative dates can be created which can be viewed by all the decision makers. That will enable the participants to make their decisions in accordance to the preferences of each other.

3 Simulation of Redesigned Process Model. In order to assess the efficiency of the transformation T, we plan to develop a set of indicators K_j and to conduct a simulation of the process model for estimating the values of these indicators. This is the final step of the improvement scheme, and, depending on the results of the simulation, the conclusion is made, either to accept the improved process model P' and start using it at the enterprise, or to conduct further improvements of the process model. Such a decision can be done, for example, by a business analyst or higher management.

4 Conclusion

In the presented paper we provided a hybrid approach for business process improvement, which consists in identifying a specific pattern in the business process model and in transforming this model in order to increase its efficiency. Also we presented several notions from the decision theory and showed how they could be correlated with the business process management. As a pattern for redesign, we introduced a special business process, the decision task, and showed that the internal structure of the decision task can be improved by maximization of the payoff of the task for the business process participants.

However, the presentation of the decision tasks and the transformation rule is not yet strictly formalized and should be done in the future. A notable limitation of our approach is that our investigation of the possibilities of the decision tasks improvement is bound to the dependencies between data attributes of the data objects of the processes at the modeling level. Nevertheless, in future we plan to enhance the approach with the execution semantics of the data objects.

As well, an extended formalization of the binding of the decision theory and business process management is planned which will allow to apply our approach to a broader class of business processes incorporating the decision making. For instance, in the current paper we referenced the business process of decision making in the business process of scheduling of the meetings, but it could be extended to the integrated time management at the enterprise. As another example of the decision tasks can serve the business process of the quality control of the product, where the evaluation is made independently by experts according to the predefined scale and the experts have to come to a compromise product assessment.

References

1. Marwa M.Essam and S. Limam Mansar. Towards a software framework for automatic business process redesign. *ACEEE International Journal on Communication*, 2(1):6, March 2011.
2. Andreas Meyer, Sergey Smirnov, and Mathias Weske. Data in business processes. *EMISA Forum*, 31(3):5–31, 2011.
3. John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
4. Preeti S. Patil, Srikantha Rao, and Suryakant B. Patil. Optimization of data warehousing system: Simplification in reporting and analysis. *IJCA Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET)*, (9):33–37, 2011. Published by Foundation of Computer Science.
5. Hajo A. Reijers and S. Liman Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005.
6. Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

Integration of Documentation Task into Medical Treatment Processes in the Hospital

Marcin Hewelt and Aaron Kunde

Hasso Plattner Institute at the University of Potsdam
{marcin.hewelt, aaron.kunde}@hpi.uni-potsdam.de

Abstract. Medical Guidelines capture medical knowledge how to diagnose and treat certain diseases. To some degree they describe patient data that is required to make treatment decisions. Usually documentation of medical procedures is carried out as an afterthought and produces additional workload for doctors and nurses. In this contribution we study the German guideline for neck pain, translate the guideline into a process model and identify the tasks of this process where documentation needs to take place, as well as the required data.

1 Introduction

Documentation processes are an essential part of medical practice in a hospital. The documented data is used for billing the health insurer, communication with other health care facilities (e.g. aftercare, rehabilitation), medical research and evaluation of treatment options. These days the documented data has not always direct benefit for the current treatment of a patient. Often only data that appears useful in the short run is captured, neglecting the requirements of retrospective research. Finally, documentation processes are also error-prone, the more so the longer the topic to be documented lies in the past. As a result, documentation of medical procedures is often carried out only as an afterthought. This leads to additional workload for doctors, nurses and technical staff as well as redundant activities and media breaks. A study performed by the Deutsches Krankenhausinstitut e.V. (DKI) found for example, that surgeons spend 2 hours for patient-centric documentation and another 42 minutes for administrative documentation a day. Internists spend more than 2,5 hours for patient-centric and 40 minutes for administrative documentation a day ([6], [1]). Administrative documentation has been defined in this study as necessary data and documents for communicating with insurance or other administrative offices. Patient-centric documentation on the other hand contains all documents having the patient in focus, like diagnosis and treatment. If documentation were better integrated with the treatment process, data produced during treatment could automatically be captured for documentation and the error rate as well as the overhead could be reduced. This way doctors and nurses would have more time for patients. In addition, if documentation of the current treatment had direct benefit for patients and medical staff during the treatment, the motivation to document would be higher.

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

Our idea is to use a formal model of the treatment process, which can be executed by a process engine. This way process guidance to the practitioner could be provided, e.g. making recommendations about the next possible treatment steps. To make an appropriate recommendation, the system needs certain information, based on data, which has been documented before. This information contains the state of the process, together with all necessary information about the patient and the environment to support the decision. Our approach is to log all the necessary information, which has lead to the decision together with the decision itself as documentation of the process. Therefore we need a formal model of healthcare processes.

The literature points at many contributions that represent a Clinical Practice Guideline (CPG) in a way usable by IT systems called a Computer Interpretable Guideline (CIG). Our idea is to first express the quite simple German CPG for neck pain with a Business Process Model and Notation (BPMN) model. Since medical treatment processes can be seen as workflows, standard process modeling languages, like BPMN [4], should be suitable to represent actual processes in a hospital as well as CPGs. Most industries employ BPMN to model processes. Even if BPMN has the same acceptence problems among healthcare professionals like other proposed formalisms, it is understood and widely accepted in other professions, e.g. by management and knowledge engineers. Hence other modelling experts can help healthcare professionals with modelling a CPG more easily.

If a CPG can be represented with BPMN it can be executed by common process engines, which are already used and established for business processes like billing and accounting. This supports the integration of CPG models in existing environments. The integrated CPG models can be used to verify existing processes or to build a Decision Support System (DSS) for treatment.

As a second step we annotate those activities in the process model which produce information that is to be documented. This is expressed via the BPMN construct of a data object. We simply list the information we are interested in documenting on the arc between the task that produces them and the data object. According to the Business Process Modeling (BPM) life cycle the process model has to be configured by attaching additional technical details in order to make it executable by a process engine. It is also during this step that the decision made at gateways are to be specified in detail.

This way, the necessary data for each executable task is specified. The data which is needed for each decision is specified as well and therefore known by the execution engine. All this data is then logged as process documentation. Furthermore, this data can be used for consistency checking in the model: Each task can check, if the data it needs is specified by some task before by building the reverse reachability graph.

2 Background

A CPG represents evidence-based medical best practices for certain medical conditions, e.g. lymphoma, describing their diagnosis and treatment. The def-

inition and importance of CPGs have been mentioned in [10]. Grimshaw and Russell analysed almost 60 published evaluations of CPG usage and found that a great majority reported a significant improvement of quality of care and patient outcome [3]. To ensure the quality of those recommendations, in Germany CPGs are reviewed and staged in a systematic way by the Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF). In this system the stage "S3" describes the highest quality. Although CPGs have a certain logical structure, they consist of free-form text, tables, and diagrams. Many of them are presented in a long and a short form. The short form may consist of only one or two pages, describing the overall procedure in a flow-chart. On the other hand, the long version may contain up to more than hundred pages describing each step in detail with possible side effects.

Since CPGs are normal text-based documents, they can not be used directly in computer systems. To do so, CPGs have to be formalized into computer-interpretable representations, so-called CIGs. There are many examples of CIGs, which have all been designed by different institutions with different goals in mind.

Wang et al. [10] analyze 11 CIG languages to identify the most important components. They found that *actions* and *decisions* are essential in all analyzed languages. Additional important components are *patient state* and *execution state*. All of the analyzed languages support this components. Peleg et al. [8] use the term Task Network Model (TNM) to describe CIGs that support hierarchical decomposition over time. This ability is an important feature during the modeling process, enabling a top-down approach and modularization. To find similarities and differences of languages that are based on TNMs, six languages are compared in [8]. Despite the fact, that each language has been developed with a different goal in mind, common components, like structuring treatment plans sequentially, which may be parallel or not, have been identified. According to this research exisiting CIGs share most of the features, however, the CIG language landscape seems very fragmented and lacks a standard.

Mulyar et al. [5] investigate the support of four TNM-based CIG languages for 43 common workflow patterns like *Multichoice*, *Milestone* or *Recursion*[1]. They conclude that of all workflow patterns only up to 22 patterns have been supported by one of the investigated languages (PROforma). On the other hand not much flexibility has been added by the CIG languages. This leads to the question, if those specialized CIG languages are really necessary, or if CPGs can be modeled with a general modeling language, like BPMN [4], which is a freely available standard by the International Organization for Standardization (ISO).

A current review [7] discusses and classifies 21 articles about research on CIGs. The classification describes 8 topics of interest for CIG research. These topics describe constitute phases in the life cycle of a CIG, like the integration with Electronical Health Record (EHR), validation or exception handling. The paper gives a good overview of the literature of the past 20 years and it becomes clear, that the development and deployment of a CIG is not trivial.

[1] from `http://workflowpatterns.com`

3 Conceptual Model

Essentially, guidelines are process models that contain medical knowledge. BPMN is a widely accepted and understood standard to model business processes. Because it allows to represent all the relevant concepts of CPGs identified in [10] and [7], BPMN seems also suitable to model and represent CPGs. A CPG modeled with BPMN could be integrated into IT systems that are already employed in the hospital to support administrative workflows. Since BPMN is an open standard, sharing of guideline models would be easier. Additionally, there are many tools for validating and transforming BPMN-models available.

To demonstrate our approach, we took the neck pain guideline created by Deutsche Gesellschaft für Allgemeinmedizin und Familienmedizin (DEGAM)[2] [2] (80 pages of narrative, two flow diagrams) and constructed an initial model in BPMN. This model in figure 1 shows a conceptual top-down view, based on the flow-chart from the guideline. We used the tasks from the original document and added necessary decision gates. Additionally the gates were annotated with the dataflow necessary for documentation.

Our goal is to use as few constructs as possible to map the necessary concepts from the medical domain. The fewer constructs a model involves, the clearer and easier to understand it is. In the following we propose a mapping from common CIG constructs, which were developed to capture concepts of medical CPGs.

Action An action is a simple task. BPMN has the notion of atomic tasks, which can be executed by the process engine. Each action can be connected to a data object in which data is stored, like an EHR. The user only has to specify which data shall be stored in which system We do not want to concern the user with details on how the data is stored.This has to be done by configuring the process engine.

Decision Decisions are made by medical experts. They regard the diagnosis to be made or the treatment to be chosen. This decisions are based on the actual patient state as represented in one or more data objects providing an EHR. A DSS might calculate ranked recommendations based on the documented data in the EHR and present it to the practitioner. The practitioner then can choose one of those options (or something completely different). Her choice is then documented in the EHR, together with an optional argument for the decision. Process models in BPMN use the construct of gateways which comes in different variants to represent choices to be taken during the process execution. For decisions we take XOR-Gateways to represent exclusive decisions.

Data inquiry Data inquiry for actions is implicit in the model. The inquired data for decisions is for now modeled using textual annotations at the gateways.

Hierarchical plans Sub-processes in BPMN support the hierarchical decomposition of processes. This construct therefore allows to decompose the model, an operation also possible in TNMs.

[2] `http://leitlinien.degam.de/index.php?id=269`

Parallel tasks BPMN supports parallelism using parallel gateways (AND-Gateways). Those can be used to model parallel tasks, like requesting multiple tests in the laboratory.

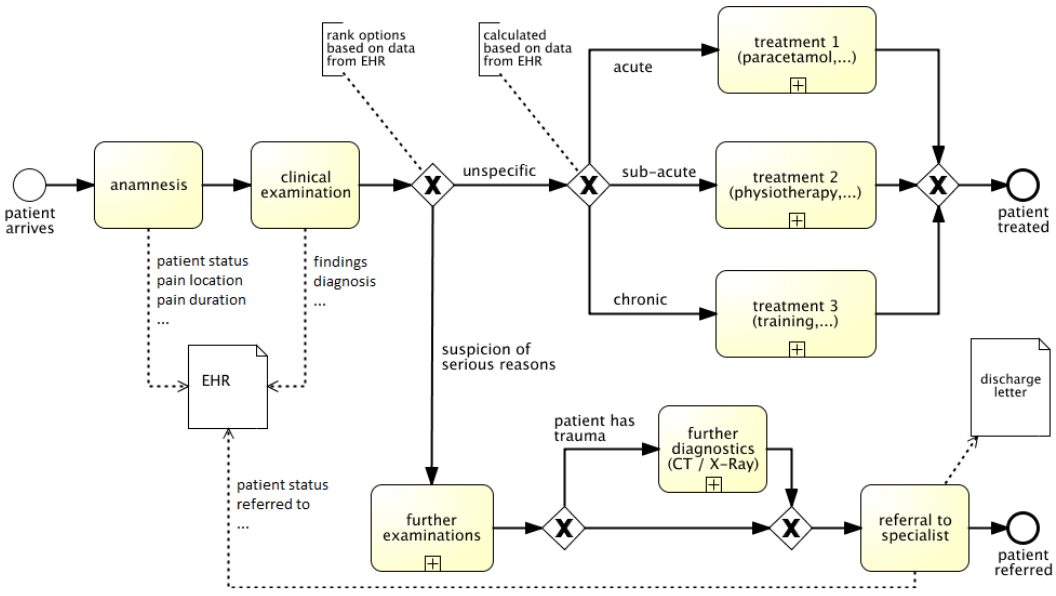


Fig. 1: BPMN process model of neck pain guideline

Scenario Neck Pain Guideline

Our model has one start event ("patient arrives") and two possible end events ("patient treated" and "patient referred"). They represent the entry of patients and the two different outcomes specified in the guideline.

The action "anamnesis" is modeled as an atomic task. During "anamnesis" the doctor asks the patient about her medical history and current condition, e.g. where the pain is located and when it began. This information needs to be documented. The documentation system is modeled as a data object called "EHR". The user input is implicit in the BPMN and does not need to be modeled explicitly. The output for the process documentation has to be modeled, but only to the point, that the user has to specify, which data has to be documented. This specification is done by writing the data fields on the edges, which are associated with the data object. In our example it is shown, that some data called "patient status", "pain location", "pain duration" and more is stored in the data object. The tasks "clinical examination" and "referral to specialist" also store data in this

data object. An additional data object “discharge letter” is created in the task “referral to specialist”. Only atomic tasks can have an association to a data object. Sub-processes like “further examinations” are not connected to data objects so as not to overload the model. Instead, if a sub-process contains atomic tasks, which consume or produce data, this is modeled in the sub-process. Because data objects are referenced by name, a sub-process can use the same data store (e.g. data object “EHR”) as the overall process, just by using the same name.

Several kinds of decisions are included in our example. The first one is the most complex. Based on the data stored in the “EHR”, a ranking is presented which gives the user to classify the patients pain either as “unspecific” or as “suspicion of serious reasons”. This is modeled for now using a textual annotation. During the configuration phase of the process model an algorithm needs to be provided by the process implementer that realizes the decision. The best option would be to devise a generic decision algorithm and a domain-specific language for modeling decisions (required data, possible options, arguments for options) which expressions could be interpreted by the algorithm. This use of user-specified annotation languages is supported by the BPMN 2.0 standard. However, this is out of scope for this contribution. The second kind of decision is shown after the task “further examinations”. The doctor can decide, if she should refer the patient to a specialist directly or perform X-Ray diagnostics before.

4 Discussion

The hospital process domain exhibits a gap between medical treatment and organizational processes. While CPGs provide valuable knowledge to medical practitioners they are not designed to be supported by IT systems. Therefore CIGs were introduced as formal representation of CPGs. However, they focus on medical aspects of the process and cannot be easily integrated with organizational aspects, like registration of examinations, or billing the health insurer. [5] found that CIGs do not add flexibility to treatment processes. On the other hand, BPMN was developed to express these business processes and, together with a process engine, to support them and integrate existing IT systems. Previous work by Reijers et al. [9] found out, that workflow management, an approach similar to BPM, can be successfully used to support medical treatment processes.

In this contribution we only consider the integration of documentation steps into the treatment process. To this end we used standard BPMN with some additional annotations at the arcs to model one specific textual CPG. Large portions of the integration work belong to the configuration phase in the BPM life cycle, in which detailed specification for the process model is provided, to make the model executable. This includes binding the data objects, used in the model, to data storages like EHR, mapping the annotations to information produced by the corresponding task, and providing the decision rules for gateways.

We plan to evaluate the open-source process engines Activiti [3], jBPM[4] & Drools[5], ruote[6], Enhydra Shark[7], camunda[8] and Bonita BPM[9] to our needs. Then we will implement the modeled guideline in one of these engines address the afore-mentioned questions.

References

1. Blum, U., Müller, K.: Dokumentationsaufwand im Ärztlichen Dienst der Krankenhäuser. das Krankenhaus pp. 544–548 (Jul 2003)
2. DEGAM: Leitline Nackenschmerzen. AWMF (Jun 2009)
3. Grimshaw, J.M., Russell, I.T.: Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations. Lancet 342(8883), 1317–1322 (Nov 1993), PMID: 7901634
4. ISO/IEC: Information technology – object management group business process model and notation. Tech. Rep. 19510, ISO/IEC (Jul 2013)
5. Mulyar, N., van der Aalst, W.M., Peleg, M.: A pattern-based analysis of clinical computer-interpretable guideline modeling languages. Journal of the American Medical Informatics Association 14(6), 781–787 (Nov 2007)
6. Müller, K., Blum, U.: Krankenhausärzte: Enormer Dokumentationsaufwand. Deutsches Ärzteblatt 23(100), 24 (Jun 2003)
7. Peleg, M.: Computer-interpretable clinical guidelines: A methodological review. Journal of Biomedical Informatics 46(4), 744–763 (Aug 2013)
8. Peleg, M., Bury, J., Fox, J., Greenes, R.A., Hall, R., Johnson, P.D., Jones, N., Kumar, A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E.H., Stefanelli, M.: Comparing computer-interpretable guideline models: A case-study approach. Journal of the American Medical Informatics Association 10(1), 52–68 (2003)
9. Reijers, H.A., Russell, N., Geer, S.v.d., Krekels, G.A.M.: Workflow for healthcare: A methodology for realizing flexible medical treatment processes. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) Business Process Management Workshops, pp. 593–604. No. 43 in Lecture Notes in Business Information Processing, Springer Berlin Heidelberg (Jan 2010), http://link.springer.com/chapter/10.1007/978-3-642-12186-9_57
10. Wang, D., Pelegb, M., Tub, S.W., Boxwalac, A.A., Greenesc, R.A., Patela, V.L., Shortliffea, E.H.: Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines:: A literature review of guideline representation models. International Journal of Medical Informatics 68(1–3), 59–70 (Dec 2002)

[3] http://activiti.org
[4] http://www.jboss.org/jbpm
[5] http://www.jboss.org/drools
[6] http://ruote.rubyforge.org/
[7] http://shark.ow2.org/doc/1.0/
[8] http://www.camunda.com
[9] http://www.bonitasoft.com/

Towards Quantifying the Adaptability of Executable BPMN Processes

Jörg Lenhard

Distributed Systems Group, University of Bamberg, Germany
joerg.lenhard@uni-bamberg.de

Abstract Process languages such as the *Business Process Model and Notation* 2.0 or the *Web Services Business Process Execution Language* promise the portability of executable artifacts among different runtime environments, given these artifacts conform to the respective specification. However, due to the natural imperfectness and differing priorities of runtime environments, actual portability of process code is often hard to achieve. A first step towards tackling this problem is the quantification of the actual *degree of portability* of process code using software metrics. The ISO/IEC 25010 software quality model defines portability as a main software quality characteristic with several sub-characteristics. One of these is *adaptability*, the degree to which a piece of software can be adapted in order to be executed in a different environment. In this paper, we propose a mechanism for quantifying the degree of adaptability of BPMN 2.0 processes and demonstrate its computation.

Keywords: Adaptability, ISO/IEC 25010, BPMN, Metrics

1 Motivation

A central part of process-aware applications is the runtime platform for executable process models. To run on such a platform, processes need to be tailored to it, thus often locking them into that particular platform. This lock-in effect is undesirable. Application *portability* addresses this issue.

A way to improve the portability of an application is by programming it according to an open specification that promises the portability of the outcome. This is the route taken for various process languages [10], such as the *Business Process Model and Notation* (BPMN) 2.0 [16] or the *Web Services Business Process Execution Language* (BPEL) 2.0 [15]. Being open international standards, these languages name portability as a central goal. The problem is that these standards are just specifications and portability of code ultimately depends on their implementations. These, however, rarely implement the complete specification and naturally inhibit faults or mandate the usage of non-standard extensions that limit the portability of a process. Huge differences in standard conformance have been demonstrated for BPEL runtimes [4, 5]. It can be expected that the situation is the same in the case of BPMN, as indicated by recent studies showing

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

compliance issues for its serialization format [3]. This implies that processes implemented in these languages, despite being conformant to an open international standard, cannot be considered as portable per se.

A first step towards tackling this problem is the ability to quantify it: to be able to compute a degree of portability, or adaptability, for a given process implemented in a particular language using software metrics. This could yield several benefits, as for instance:

1. When integrated into a metrics suite, developers could continuously inspect the portability or adaptability of the process during development. This would allow them to get direct feedback for changes they introduce and make them aware of changes that limit portability or adaptability [13]. Raising their awareness has the potential to result in more portable code.
2. When having to port a process, metrics can be used as a basis for decision making. A high degree of portability or adaptability translates to a high likelihood that the process can actually be ported or adapted. In contrast to this, a low degree can support the decision to rewrite the process from scratch for the new platform.
3. When selecting one among a set of alternative processes for execution, metrics can serve as a means for quality comparison and ranking the alternatives.

As a basis for such a quantification, the ISO/IEC 25010 software quality model [7] can be of help. This new revision of the widely-accepted ISO/IEC 9126 quality model lists portability as a main quality attribute of software, consisting of several sub-attributes: *Adaptability*, *installability*, and *replaceability*. Each of these characteristics should be measurable to compute the degree of portability and it is our goal to build a measurement framework that achieves this for process-aware and service-oriented systems. Since we addressed direct code portability [12] and installability [11] in previous work, we now try to tackle the quantification of adaptability for this type of software. In this paper, we try to quantify the adaptability of BPMN processes using structural code metrics. To meet this end, we propose a mechanism for computing such metrics and demonstrate its application in a use case. We are trying to compute a quantitative representation of the likelihood that the code of a process can be adapted to a different form that results in the same runtime behavior. We are *not* trying to provide a metric that states if a process can be modified to run on a particular engine.

We have to emphasize that the purpose of this paper is the proposal and description of the mechanism and metrics. Due to this scope and the page limit, we defer the important aspect of the validation of the metrics, for instance in terms of measurement theory [2] or construct validity [9], to future work.

The remainder of the paper is structured as follows: In the next section, we discuss related notions of adaptability and adaptability metrics. In section 3 we introduce our approach and explain our proposed metrics. Thereafter, we evaluate the approach by computing the adaptability degree for a use case process. Finally, we draw a conclusion and point to future work.

2 Notions of Adaptability and Related Work

The ISO/IEC quality model defines adaptability as the "*degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments*" [7, p. 15]. Here, we focus on adaptions to the software environment only. The scenario we have in mind is a required change of a set of processes to a different runtime engine. If the processes cannot be ported directly, they have to be adapted to preserve their executability. This is different to other views of adaptability, as for instance in autonomous systems, where adaptability refers to the ability of the system to automatically cope with changing situations, such as an increased load, at runtime [17] or adapter synthesis, where adaptability refers to whether an adapter for a pair of services can be created [19].

In this paper, we try to quantify adaptability in an abstract, runtime-independent fashion. Hence, we base the following metrics on the BPMN specification [16] only. Nevertheless, it might be worthwhile to consider actual process runtimes in the computation of adaptability metrics, as for instance done in [12], since the adaptability of a particular process to a particular runtime ultimately depends on the runtime to which it should be ported. However, our aim is to allow for the measurement of adaptability already at a point in time, where no new runtime has been selected yet. Still, we plan to evaluate the usage of data on language support in runtimes to see if it can enhance the metrics proposed here.

The metrics for adaptability of the new ISO/IEC quality model [8] are not yet publicly available, but will likely be similar to that of previous versions, e.g. [6]. These metrics are based on counting the number of program functions that seem to be adaptable to different contexts. This number is contrasted with the number of functions that are required to be adaptated in the current situation, which is typically all program functions that need to be available in the new environment after porting. By relating these two numbers, one can obtain the percentage of program functions that can be adapted and thus provide a basic notion of adaptability for the complete program. However, such a measure is very coarse and there is no description of how to actually determine if a function is adaptable or not. Here, we try to provide a mechanism to determine if a program element is adaptable. Other studies that evaluate adaptability [1] rely on surveys of stakeholders. Metrics based on human judgment can have limitations in terms of reproducibility and reliability, which is why we aim to provide structural code metrics that can be computed automatically and are reproducible instead.

Such structural metrics do primarily exist for the architectural layer of a software product and not the concrete source code [17, 18]. There, adaptability is first quantified in a binary or weighted fashion for an atomic element of the respective system, such as a component in the software architecture. These element adaptability scores are then subsequently aggregated using different adaptability indices at different layers of abstraction to arrive at a global value of adaptability for the complete software architecture. This way of computing adaptability should also work when looking at code artifacts and not architectural elements of a program. Here, we focus on executable service-based processes and

try to reproduce the adaptability computation in the above sense. Thus, our idea is to quantify adaptability at the level of an atomic process element, such as an activity, and to aggregate this to a global degree for the complete process.

3 Measuring Structural Adaptability

In the terms of BPMN, an atomic process element is an activity, task, or gateway. Using the above approach, we need to assign an adaptability score to each of those elements. Our idea is to count the number of alternative representations for the functionality provided by the element that result in the same runtime behavior. In BPMN, there are typically multiple alternatives for each process element that can result in identical process behavior at runtime. The more alternatives exist for a given process element, the easier it is to replace this element with such an alternative, and hence the more adaptable the resulting code actually is.

A simple example for multiple alternative implementations of the same functionality in BPMN is repetitive execution of a task through a *Loop* marker for the task. Any of the following language constructs can be used to define repetitive execution of a task and hence can be used as an alternative to a *Loop* marker:

1. A combination of an *Exclusive Gateway* and *Sequence Flows*
2. Enclosing the task in a *Loop Sub-Process*
3. Enclosing the task in an *Ad-Hoc Sub-Process*
4. Enclosing the task in an *Event Sub-Process*

It is likely that a BPMN engine will only support a subset of these options. For instance the Activiti engine[1], currently does not support normal *Loop* markers (`standardLoopCharacteristics`). It does support the combination of *Exclusive Gateways* and *Sequence Flows*, as well as *Event Sub-Processes*, but no *Loop* or *Ad-Hoc Sub-Processes*. Given a process with a task that uses a *Loop* marker needs to be ported to the Activiti engine, the code needs be adapted to one of the versions Activiti supports. To summarize the above discussion, the adaptability score of a task with a *Loop* marker is equal to four.

3.1 Adaptability of Atomic Process Elements

We define the adaptability score for atomic process elements as:

$$AS(e) = \mid \{alt_1^e, \ldots, alt_n^e\} \mid \tag{1}$$

The *adaptability score AS* of element e is equivalent to the cardinality of the set of alternatives $\{alt_1^e, \ldots, alt_n^e\}$ for the element that are available in the language. For the approach to work, such a score must be provided for every relevant atomic element of the BPMN specification. At the moment, we are fixing the appropriate score for every element, being activities (*Tasks*, *Sub-Processes*, *Call Activities*), *Data Items*, *Events* and *Gateways*.

[1] For more information, see the Activiti user guide: `http://www.activiti.org/userguide/index.html`.

The decision on what counts as a relevant atomic element is a design choice of the approach. It is reasonable to exclude a certain set of language elements from the computation. On the one hand, these are elements that are very basic and also very common and, as a consequence, are supported by every implementation of the standard. The inclusion of these elements in the adaptability computation would only have a distorting effect. On the other hand, certain elements are simply irrelevant to process execution and their implementation in a runtime is unimportant. *Lanes* fall into the latter category, as they have no real impact on the executability of a process, but are mainly relevant to visualization. The first category contains *Sequence Flows* and *Exclusive Gateways*. *Sequence Flows* are very basic language elements and it is hard to build processes in BPMN without using them. *Ad-Hoc Processes* in combination with *Data Inputs* and *Data Outputs* can, to a limited degree, replace *Sequence Flows*, but fail for instance when parallelism is involved. As they are typically very frequent, but cannot really be adapted anyway, we exclude them from the computation. *Exclusive Gateways* can be adapted to most other forms of gateways, such as an *Inclusive Gateway* where only one expression will evaluate to true, a *Complex Gateway*, or an *Event-Based Exclusive Gateway*. Nevertheless, they are the most basic mechanism for controlling the program flow and normally available in every implementation of the specification. Including them in the computation would introduce noise into the metric value. To decide if an element belongs to the basic subset, it could be helpful to look at its typical frequency in process models. [14] disusses this aspect for an older revision of BPMN and an updated study focused on executable processes might be worthwhile.

Finally, it is important to note that this approach rewards the availability of multiple equivalent constructs in a language. From a usability point of view, this is often considered as a drawback of the language, because its users can be confused on what syntax is best to be used. However, from the viewpoint of adaptability, it is positive, since multiple alternatives increase the likelihood of having at least one of them available in a given runtime.

3.2 Aggregation of Adaptability Scores

Based on atomic adaptability scores, we now need a mechanism for aggregating these scores to a global adaptability degree for the complete process. This is necessary to allow for the comparison of different processes in terms of their adaptability. Moreover, the aggregated degree should be normalized with respect to the size of the process, to enable the comparison of processes of different size. A straightforward way of aggregating adaptability scores is the following:

1. Normalize the score for every element.
2. Similar to [18], compute the mean score of all elements in the process.

This leads to the question of how to normalize scores on an atomic level. We propose to divide the score by a reference value. This reference value can be identified by the maximum adaptability score achieved by any of the elements in the language. That way, the most adaptable language element will have a

normalized score of one, whereas other elements will have a value between zero and one. This results in the following equation:

$$AD(p) = \overline{AD(e_1, \ldots, e_n)} = \overline{(AS(e_1)/R), \ldots, (AS(e_n)/R)} \tag{2}$$

The *adaptability degree AD* of process p, which consists of the elements $e_1, \ldots, e_n$, is equal to the arithmetic mean of the *adaptability scores AS* for every element e divided by the *reference value R*.

For the choice of the reference value, which we currently determined to be six, different schemes are possible. The scheme we use here has several advantages with respect to the computation:

1. The resulting metric value always ranges in the interval of $[0, \ldots, 1]$ and thus resembles a percentage value. This scale is easy to understand and interpret, which is critical for the adoption of the metric.
2. The reference value is identical for processes of the same language. Using a reference value that is specific to a concrete process might output a more meaningful adaptability degree for that process, but it would no longer be directly comparable with different processes. That way, the metric would lose one of its primary purposes.

4 Use Case

In the following, we use an example process[2], depicted in Figure 1, to demonstrate the computation of the adaptability degree. The process consists of a *Lane*, two

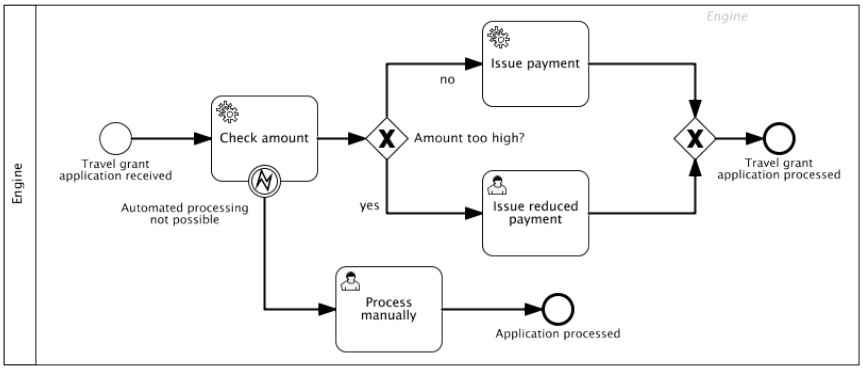


Figure 1. Travel grant application process

User Tasks and two *Service Tasks*, one of which has an *Interrupting Error Boundary Event*, two *Exclusive Gateways*, several *Sequence Flows*, as well as two *End Events* and a *Start Event*. Table 1 shows the adaptability scores we

[2] The process is executable on Camunda BPM 7.0.0. The code and instructions on how to execute it are available at `https://github.com/uniba-dsg/zeus2014`.

Table 1. Adaptability scores of process elements rounded to two decimal places

Element	*None Start Event*	*None End Event*	*Task*	*Error Boundary Event*
$AS(e)$	5	4	5	6
$AD(e)$, $R = 6$	0.83	0.67	0.83	1

determined for the respective elements of the travel grant application process. As discussed in section 3.1, *Lanes*, *Sequence Flows*, and *Exclusive Gateways* are not considered in the computation. The BPMN specification lists seven different triggers for process start, and hence seven different types of *Start Events* [16, pp. 240/241] do exist. All except for the *Timer Event* are a suitable alternative for the *None Start Event* used in the process, resulting in an adaptability score of five. For *End Events*, nine different types do exist [16, pp. 247–249], five of which, including the *None End Event*, can be used to express orderly termination, resulting in four alternatives for it. Furthermore, there are seven different types of *Tasks* in BPMN [16, pp. 158–165]. Again, the idea is that *Tasks* which are not supported by an engine can be adapted to a different type of *Task*, for instance a *Service Task* could be adapted to a *Script Task*. All tasks actively perform an action, except for the *Receive Task* which is waiting for an action, so there are five alternatives for the tasks used in the process. If the *Error Boundary Event* is not supported, it might be possible to use a different interrupting boundary event which also changes the normal flow into an exception flow. Here, a *Message*, *Escalation*, *Conditional*, *Signal*, *Multiple*, or *Multiple Parallel Interrupting Boundary Event* could achieve the same result [16, pp. 254–257]. The reference value R is six, which results in the adaptability degree values depicted in Table 1. The resulting adaptability degree for the use case is computed in the following: $AD(p) = ((1 * AD(StartEvent)) + (4 * AD(Task)) + (2 * AD(EndEvent)) + (1 * AD(ErrorEvent)))/8 = ((1 * 0.83) + (4 * 0.83) + (2 * 0.67) + (1 * 1))/8 = 0.81$.

5 Conclusion

In this paper, we proposed a mechanism for computing a degree of structural adaptability for BPMN processes that aims to quantify how easily a process can be adapted to a different form with the same runtime behavior. Furthermore, we demonstrated its computation in a use case. Such a degree can be helpful for quality assessment during development or decision support during migration.

Several aspects of the computation are still open: First, adaptability scores need to be fixed for every relevant element and they should be confirmed in peer review. Moreover, a validation of the proposed adaptability metrics is needed. On the one hand, validation should be considered from a theoretical point of view, for instance by clarifying the measurement-theoretic properties of the metrics or by confirming construct validity. On the other hand, the practical applicability of the metrics should be confirmed, for instance in an experiment with real-world processes. This could be used to evaluate if the adaptability degree can meaningfully discriminate between processes of different quality.

References

1. Aldris, A., Nugroho, A., Lago, P., Visser, J.: Measuring the Degree of Service Orientation in Proprietary SOA Systems. In: 7th IEEE International Symposium on Service-Oriented System Engineering. San Francisco Bay, USA (March 2013)
2. Briand, L., Morasca, S., Basily, V.: Property-based software engineering measurement. IEEE Transactions on Software Engineering 22(1), 68–86 (1996)
3. Geiger, M., Wirtz, G.: BPMN 2.0 Serialization - Standard Compliance Issues and Evaluation of Modeling Tools. In: 5th Int. Workshop on Enterprise Modelling and Information Systems Architectures. St. Gallen, Switzerland (September 2013)
4. Harrer, S., Lenhard, J., Wirtz, G.: BPEL Conformance in Open Source Engines. In: IEEE SOCA. Taipei, Taiwan (December 17-19 2012)
5. Harrer, S., Lenhard, J., Wirtz, G.: Open Source versus Proprietary Software in Service-Orientation: The Case of BPEL Engines. In: 11th International Conference on Service Oriented Computing (ICSOC). pp. 99–113. Berlin, Germany (2013)
6. ISO/IEC: Software engineering – Product quality – Part 3: Internal metrics (2003), 9126-3:2003
7. ISO/IEC: Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011), 25010:2011
8. ISO/IEC: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality (2013), 25023
9. Kaner, C., Bond, W.: Software Engineering Metrics: What Do They Measure and How Do We Know? In: 10th International Software Metrics Symposium. Chicago, USA (September 2004)
10. Khalaf, R., Keller, A., Leymann, F.: Business processes for Web Services: Principles and applications. IBM Systems Journal 45(2), 425–446 (2006)
11. Lenhard, J., Harrer, S., Wirtz, G.: Measuring the Installability of Service Orchestrations Using the SQuaRE Method. In: IEEE SOCA. IEEE, Kauai, Hawaii, USA (December 16-18 2013)
12. Lenhard, J., Wirtz, G.: Measuring the Portability of Service-Oriented Processes. In: 17th IEEE EDOC. Vancouver, Canada (September 2013)
13. Letouzey, J.L., Ilkiewicz, M.: Managing Technical Debt with the SQUALE Method. IEEE Software 29(6), 44–51 (2012)
14. zur Muehlen, M., Recker, J.: How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Advanced Information Systems Engineering (CAiSE). Montpellier, France (June 2008)
15. OASIS: Web Services Business Process Execution Language (April 2007), v2.0
16. OMG: Business Process Model and Notation (BPMN) Version 2.0 (January 2011)
17. Perez-Palacin, D., Mirandola, R., Merseguer, J.: On the Relationships between QoS and Software Adaptability at the Architectural Level. Journal of Systems and Software 87(1), 1–17 (2014)
18. Subramanian, N., Chung, L.: Metrics for Software Adaptability. In: Proc. Software Quality Management. Loughborough, UK (April 2001)
19. Zhou, Z., Bhiri, S., Zhuge, H., Hauswirth, M.: Assessing Service Protocol Adaptability Based on Protocol Reduction and Graph Search. Concurrency and Computation: Practice and Experience 23(9), 880–904 (2011)

Introducing Configurability into Scenario-Based Specification of Business Processes

Robert Prüfer and Jan Sürmeli

Humboldt-Universität zu Berlin, Germany
{pruefer|suermeli}@informatik.hu-berlin.de

Abstract Process model configuration is an approach to model highly similar variants of a process. In a configurable process model, events can be hidden or blocked to characterize variants. However, it may be difficult to model large processes consisting of many interacting units. To this end, one may use scenario-based specification, specifying the process in comprehensible, reoccurring parts that describe interactional behavior. In this paper, we take a look at how configurability and scenario-based specification could be merged in one approach. We particularly focus on the impact of permitting events in scenarios to be hidden or blocked.

Keywords: Business Process Modeling, Process Configuration, Scenario-based Specification

1 Introduction

Business process modeling enables design, analysis, and optimization of existing and new processes. One approach is to start with a generic reference model, and then to refine the model iteratively until the desired level of detail is reached. During refinement many highly similar variants of the process arise. To capture all these variants, the modeler could create many similar models. This immediately leads to problems regarding the maintenance and refactoring of these models.

Process configuration [1,4,12] proposes to integrate all variants of a process in one single model, marking all possible variation points. A *configurable process model* M represents a finite set $m_1, \ldots, m_n$ of highly similar process models. Each model m_i is the result of *configuring* M with a *configuration* c_i: A configuration interprets each variation point in M and thus yields a refined process model. Current approaches for configurable process models concentrate on classical formalisms to model concurrent processes such as workflow nets [2]. Such a process model captures the complete interaction of all units carrying out the process, e.g., people, web services, and information systems. Thus, it is difficult to model large processes, and to understand these models. Modeling each unit separately produces smaller models, and facilitates the analysis and implementation of each unit. However, the interaction of all units may still be hard to assess.

Scenario-based specification tackles this problem: A *scenario* describes a part of the interaction of many units. A specification consists of a set of scenarios,

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

covering all desired interactions. Well-established scenario-based specification techniques include *High Level Message Sequence Charts* (HMSCs) [14] and *Live Sequence Charts* (LSCs) [13]. For some formalisms, there exist techniques to automatically derive a process model for each unit, bridging the scenario-based view with the classical view on distributed systems, facilitating the reuse of analysis techniques, and the implementation of each unit.

Our overall goal is to connect the clarity and intuitiveness of scenario-based specification with the advantages of configurable process models. In this paper, we discuss how the core concepts of process configuration may be introduced for scenario-based specifications. As in [3], we permit the three configuration options *allowing*, *blocking*, and *hiding* to be assigned to an event. We further propose possible meanings of these configuration options and compare them to the semantics of configurable workflow net models in [4].

We proceed as follows: We propose a syntax and discuss a possible semantics of configurable scenario-based specifications in Sect. 2. Afterwards, we discuss related work in Sect. 3. Finally, we conclude our paper and sketch possible future work in Sect. 4.

2 Configurable scenario-based specifications

In this section, we propose a syntax (Sect. 2.2) and discuss *possible semantics* (Sect. 2.3) for *configurable specifications*. First, we recall syntax and semantics of scenario-based specifications (Sect. 2.1). We mostly forgo formal definitions, and describe the concepts by means of the following running example adapted from [3]. We consider a business trip application process, roughly consisting of the following steps: Either an employee or a secretary prepares a form, which is then submitted to an administrator. Concurrently to the preparation, the employee arranges the travel. Upon receipt of a form, an administrator may either approve or reject the form, or request for changes. In the latter case, the employee updates the form, again arranges the travel and resubmits.

2.1 Syntax and semantics of scenario-based specifications

We informally recall the syntax and semantics of *distributed Life Sequence Charts* (dLSCs), a scenario-based specification language introduced in [10].

A *partially ordered run* (*run*, for short) is a set of events, partially ordered by *causality*. To each event e an *activity* $\alpha(e)$ is assigned. Figure 1 shows two runs: A labeled box represents an event e with its assigned activity $\alpha(e)$. The arrows model the causality relation. In run w_1, first a start-event occurs causing an EArrange-event and an EPrepare-event. Once both events have occurred, an ESubmit-event occurs, causing an AApprove-event. In the following, the notions of *predecessor* and *successor* always refer to the causality relation. A *prefix* is a predecessor-closed set of events, a *suffix* is a successor-closed set of events.

A *scenario* is a finite run r distinctly partitioned into its *prechart* and its *mainchart*. The prechart contains at least the minimal events of r, that is, the

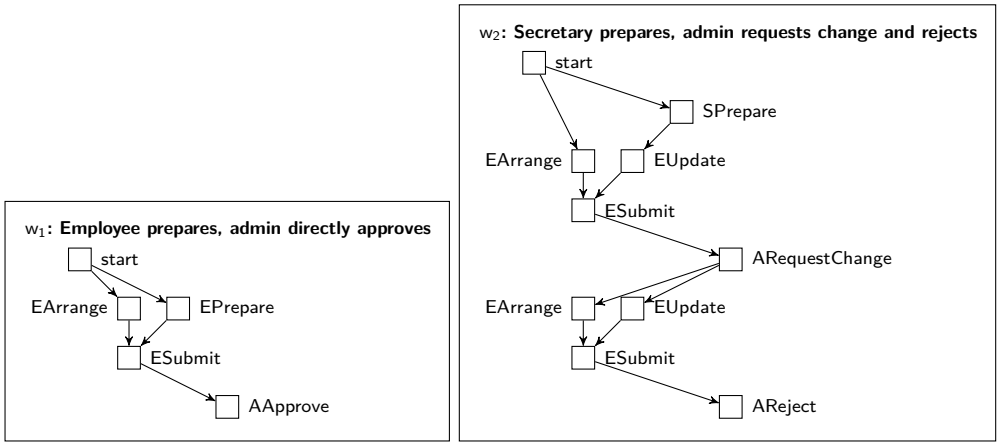


Figure 1: Two runs w_1 and w_2 of specification S in Fig. 2

events without predecessors. The main charts contains the remaining events. Graphically, we separate prechart and main chart by a dashed line. Ignoring the additional annotations in curly brackets, Fig. 2 shows six scenarios $\mathsf{s}_1, \ldots, \mathsf{s}_6$. A *specification* S is a finite set of scenarios together with an *initial run* i. The scenarios $\mathsf{s}_1, \ldots, \mathsf{s}_6$ and the initial run i in Fig. 2 form the specification S.

The semantics of a specification S is the set $\mathcal{R}(S)$ of its *maximal runs*. We first introduce the notion of a *run of* S, then we define the notion of *maximal* runs. Intuitively, we construct runs as follows: We begin with the initial run, and subsequently append the main chart of a scenario whose prechart matches the suffix of the currently constructed run. Formally, we define the notion of a run of S recursively. As base case, the initial run of S is a run of S. Let w be a run of S, and s be a scenario of S, such that the prechart of s is a suffix (up to isomorphism) of w. Then, appending the main chart of s to w yields a run of S.

A run w of S is *maximal*, if it is not a prefix of any other run of S. That is, there is no prechart of a scenario in S which is a suffix (up to isomorphism) of w. As an example, in Fig. 1, w_1 starts with i. As i is isomorphic to the prechart of s_2, we append the main chart of s_2. The prechart of s_4 is now isomorphic to a suffix of the current run. We append s_4, yielding w_1. Similarly, we construct w_2 by appending the main charts of s_1, s_3, s_6, s_3, and s_5 to i. Both runs are maximal, therefore $\mathsf{w}_1, \mathsf{w}_2 \in \mathcal{R}(\mathsf{S})$.

2.2 Syntax of configurable specifications

We consider the *configuration options* of *allowing*, *blocking*, and *hiding*, denoted by $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{H}$, respectively. According to [3], *allowing* means to not change behavior, *blocking* removes the event together with all its successors, and *hiding* an event means to skip it while preserving remaining behavior.

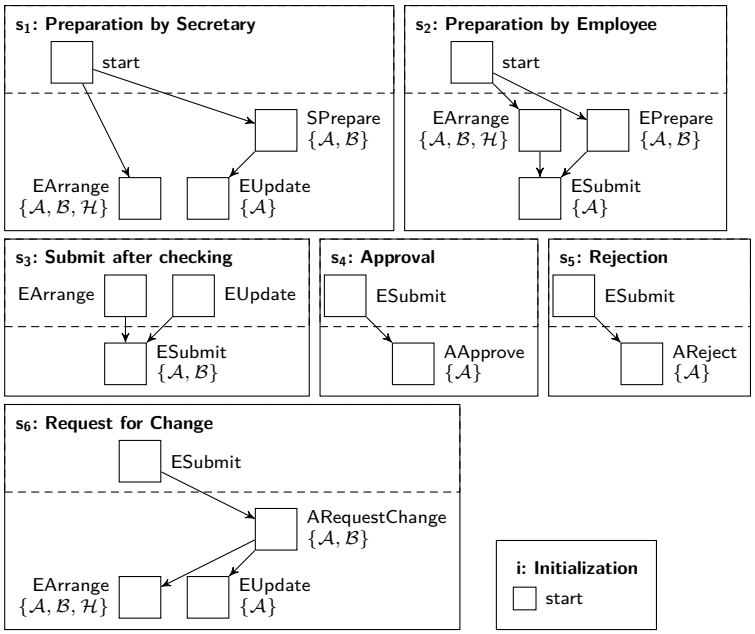


Figure 2: A *configurable specification* (S, C) consisting of the specification S with scenarios $\mathsf{s}_1, \ldots, \mathsf{s}_6$ and initial run i; C is defined by annotations in curly brackets.

To make a specification *configurable*, one chooses a set of configuration options for each event in a main chart. Thus, a *configurable specification* (S, C) is a specification S together with a function C mapping each event e of a main chart of a scenario in S to its *configuration options* $C(e) \subseteq \{\mathcal{A}, \mathcal{B}, \mathcal{H}\}$. In the following, we restrict ourselves to configurable specifications where $\mathcal{A} \in C(e)$ for each event e. We depict C as annotations to events. We omit $\mathcal{A}$, and if $C(e) = \{\mathcal{A}\}$, we completely omit the annotation. Figure 2 shows a configurable specification (S, C). In the following example, we write $x.y$ for the y-event in scenario x. E.g., $\mathcal{A}$ and $\mathcal{B}$ are the configuration options of $\mathsf{s}_1.\mathsf{SPrepare}$ in Fig. 2.

A *configuration* c of a configurable specification (S, C) is a function mapping each event e of a main chart of S to $c(e) \in C(e)$. That is, a configuration chooses a configuration option for each event. We observe that c is properly determined by its blocked and hidden events. Thus, the following is a well-defined configuration of (S, C) in Fig. 2: $\mathsf{c}_1 = \{\mathsf{s}_1.\mathsf{SPrepare}, \mathsf{s}_3.\mathsf{ESubmit}, \mathsf{s}_6.\mathsf{ARequestChange} \mapsto \mathcal{B}\}$. In contrast to that, $\mathsf{c}_2 = \{\mathsf{s}_1.\mathsf{SPrepare}, \mathsf{s}_1.\mathsf{EUpdate}, \mathsf{s}_1.\mathsf{ARequestChange} \mapsto \mathcal{B}\}$ is not a configuration of (S, C), because $\mathsf{c}_2(\mathsf{s}_1.\mathsf{EUpdate}) = \mathcal{B} \notin \mathsf{C}(\mathsf{s}_1.\mathsf{EUpdate})$.

2.3 Semantics of configurable specifications

We propose the semantics of a configurable specification (S, C) together with a configuration c of (S, C) to be a specification $[\![(S, C)]\!]_c$. Thus, (S, C) represents

a set of specifications. Following the semantics in [3], we propose that *allowing* an event does not interfere with its occurrence. Therefore, allowing each event in (S, C) yields S. In the following, we say that c *introduces* or *removes* behavior, if $\mathcal{R}(\llbracket (S,C) \rrbracket_c) \setminus \mathcal{R}(S) \neq \emptyset$ or $\mathcal{R}(S) \setminus \mathcal{R}(\llbracket (S,C) \rrbracket_c) \neq \emptyset$, respectively. In the remainder, we separately discuss possible semantics of *blocking* and *hiding*.

Blocking events. Intuitively, a blocked event and all its successors must not occur. We can think of two semantics $\mathcal{B}_{\text{full}}$ and $\mathcal{B}_{\text{part}}$ for blocking an event e in a scenario s: Under $\mathcal{B}_{\text{full}}$-semantics, the whole scenario s cannot occur and therefore is removed from the specification. Under $\mathcal{B}_{\text{part}}$-semantics e and all its successors are removed from s. For example, consider scenario $\mathsf{s_1}$ in Fig. 2 and example configuration $\mathsf{c_1}$ from Sect. 2. According to $\mathcal{B}_{\text{full}}$-semantics, $\mathsf{s_1}$ would be removed. According to $\mathcal{B}_{\text{part}}$-semantics, only the events $\mathsf{s_1}.\mathsf{SPrepare}$ and $\mathsf{s_1}.\mathsf{EUpdate}$ would be removed from $\mathsf{s_1}$. We observe that both semantics $\mathcal{B}_{\text{full}}$ and $\mathcal{B}_{\text{part}}$ in general *remove* behavior: For example, under each of both semantics, $\mathsf{w_2}$ in Fig. 1 is not a run of $\llbracket (\mathsf{S}, \mathsf{C}) \rrbracket_{\mathsf{c_1}}$ although it is a run of S. However, the $\mathcal{B}_{\text{part}}$-semantics has an additional impact: Removing an event with its successors in a scenario in general also *introduces* behavior: The construction of a run may yield prefixes, and thus may even allow to append new scenarios.

Hiding events. According to [3], *hiding* an event e intuitively means to skip it while *preserving* all other behavior. Especially, the successors of e still can occur. Technically, $\alpha(e)$ is set to τ, neither removing nor introducing behavior.

The crucial point with hiding events in scenarios is the different *precondition* for the occurrence of an event and of a scenario, respectively. In [3], the occurrence of an event is determined by the *state* of the process. As changing the activity of an event e has no impact on the state reached, hiding does not influence occurrence of any event. In contrast, occurrence of a scenario is determined by its prechart, i.e., whether a partial order of activities occurred. Therefore, changing the activity of an event in some scenario s can influence whether another scenario s' can occur. Thus, changing the activity of an event can introduce or remove behavior.

Consequently, it must be discussed how $\llbracket (S, C) \rrbracket_c$ can be defined such that behavior is preserved. This could range from simple changes in single precharts to the insertion of new scenarios. An elaborate discussion of this definition is out of the scope of this paper and is left for future work.

3 Related work

We chose *distributed life sequence charts* (dLSCs)[10] as underlying formalism for scenarios, because (1) dLSCs are based on partially ordered runs of events, which easily allow to add concepts of configurability to distributed systems, and (2) there exist techniques [9,10] to synthesize distributed components out of a dLSC specification. Additionally, dLSCs adopt the concepts of *prechart* and *main chart* from LSCs [13] for composition of scenarios. Hence, a single scenario in form of a

dLSC describes a *self-contained story*, which may be advantageous compared to the automata-based composition mechanism of HMSCs [14] as discussed in [11].

Similar to the scenario-based approach, the idea of *Process Fragments* [8] is to model small pieces of a process and to compose them. In contrast to scenarios, in this approach processes are assumed to be acyclic. Further, the approach does not enable the modeler to specify a distinguished precondition for occurrence of a process fragment. To compose two process fragments p_1 and p_2, it must be explicitly specified which activities of p_1 are to be connected to which activities of p_2 [7].

The configuration options – allowing, blocking and hiding of events – are introduced in [4]. In [3,4], the authors tackle the problem of finding and characterizing the set of all configurations leading to a *behaviorally correct* process model. In [6], the authors describe techniques to *discover* a configurable process model from an event log. In [15], the authors describe how configurable process models may be created by *merging* process models. Both discovery and merging thus can be seen as alternative approaches yielding configurable process models. As an alternative to creating a configurable process model, an approach to improve an existing reference model is described in [16]. For an overview of approaches to cope with variability in Business Processes, see [5].

In *Software Product Lines Engineering (SPLE)* [17], a *feature model* represents variants of a product. Whereas the main purpose of a feature model is to characterize valid product lines, modeling variants of a business process aims at verifying *behavioral* properties of this process.

4 Conclusion and future work

In this paper, we proposed syntax and possible semantics of configurable scenario-based specifications as a method to model variants of a process. We adopted an existing approach for configurable process models [4] to the scenario-based specification formalism of dLSCs [10]. As a proper semantics for hiding needs to be discussed carefully, this is an immediate starting point for future work. We restricted ourselves to a pure control flow view of a process. We believe that it is interesting to investigate *data-dependent configurability*. The approach to integrate data in scenarios in [11] could serve as a useful formal basis. This approach also introduces *abstraction*, allowing to specify optional behavior, which is interesting in combination with hiding. Further, a method to synthesize a configurable process model out of a configurable specification would allow to reuse techniques from [3] to characterize behaviorally correct process models. Whereas configurable process models have been assessed in case studies such as [12], the proposed formalism in this paper still needs evaluation. Here, we plan a case study in the healthcare sector. We believe that this is a reasonable application domain, because (1) healthcare processes consist of different actors performing complex interactional behavior, and (2) as individual treatment of patients inherently leads to different variants of one process, treatment of a significant number of patients could be captured by a configurable process model.

References

1. Process configuration, http://www.processconfiguration.com
2. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: Classification, decidability, and analysis. Form. Asp. Comput. 23(3), 333–363 (May 2011)
3. van der Aalst, W.M.P., Lohmann, N., La Rosa, M.: Ensuring correctness during process configuration via partner synthesis. Inf. Syst. 37(6), 574–592 (2012)
4. van der Aalst, W., Dumas, M., Gottschalk, F., ter Hofstede, A., La Rosa, M., Mendling, J.: Preserving Correctness During Business Process Model Configuration. Formal Aspects of Computing (2010)
5. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Enhancing modeling and change support for process families through change patterns. In: Nurcan, S., Proper, H.A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T.A., Bider, I. (eds.) BMMDS/EMMSAD. Lecture Notes in Business Information Processing, vol. 147, pp. 246–260. Springer (2013)
6. Buijs, J., Dongen, B., van der Aalst, W.: Mining configurable process models from collections of event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management, Lecture Notes in Computer Science, vol. 8094, pp. 33–48. Springer Berlin Heidelberg (2013)
7. Eberle, H., Leymann, F., Schleicher, D., Schumm, D., Unger, T.: Process fragment composition operations. In: APSCC. pp. 157–163. IEEE Computer Society (2010)
8. Eberle, H., Unger, T., Leymann, F.: Process fragments. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM Conferences (1). Lecture Notes in Computer Science, vol. 5870, pp. 398–405. Springer (2009)
9. Fahland, D.: From Scenarios To Components. Ph.D. thesis, Humboldt-Universität zu Berlin (2010), http://repository.tue.nl/685341
10. Fahland, D., Kantor, A.: Synthesizing decentralized components from a variant of live sequence charts. In: Hammoudi, S., Pires, L.F., Filipe, J., das Neves, R.C. (eds.) MODELSWARD. pp. 25–38. SciTePress (2013)
11. Fahland, D., Prüfer, R.: Data and Abstraction for Scenario-Based Modeling with Petri Nets. In: Haddad, S., Pomello, L. (eds.) Petri Nets 2012. Lecture Notes in Computer Science, vol. 7347, pp. 168 – 187. Hamburg, Germany (June 2012)
12. Gottschalk, F., Wagemakers, T., Jansen-Vullers, M., van der Aalst, W., La Rosa, M.: Configurable process models: Experiences from a municipality case study. In: Proceedings of the 21st International Conference on Advanced Information Systems (CAiSE 09). Lecture Notes in Computer Science, vol. 5565, pp. 486–500. Springer Verlag, Berlin Heidelberg (June 2009)
13. Harel, D., Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2003)
14. ITU-T: Message Sequence Chart (MSC). Recommendation Z.120, International Telecommunication Union, Geneva (2004)
15. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Business process model merging: An approach to business process consolidation. ACM Trans. Softw. Eng. Methodol. 22(2), 11:1–11:42 (Mar 2013)
16. Li, C., Reichert, M., Wombacher, A.: The minadept clustering approach for discovering reference process models out of process variants. International Journal of Cooperative Information Systems 19(03n04), 159–203 (2010)
17. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering - Foundations, Principles, and Techniques. Springer (2005)

Supporting Informal Processes

C. Timurhan Sungur, Oliver Kopp, and Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

Abstract People play an indispensable role in many tasks in various domains and they collaborate to accomplish those tasks. During these collaborations software tools are used, data is created/consumed and best practices might be applied. These a priori unknown informal processes are conducted with the help of experience of their actual performers. In this work, a new concept of supporting these informal processes will be introduced, i.e., *Informal Process Support Model*, consisting of *Informal Process Essentials* and *Informal Process Recommendations*, which support informal processes based on the previous executions without limiting their flexibility. Furthermore, we will introduce how these concepts can be realized with the use of Topology Orchestration Specification for Cloud Applications (TOSCA).

Keywords: ad-hoc processes, informal processes, informal processes support model, informal process essentials, informal process recommendations

1 Introduction

In various domains, e.g., manufacturing, scientific, IT, etc., business process (aka. workflow) models are used for documenting the implicit knowledge, re-use and for automation purposes. In these models, domain experts predefine the actual execution steps for the enactment of the corresponding process. Beside these formal processes, there are informal processes which are typically human-centric and carried out based on the experience of their human performers. These processes are called informal processes on account of the lack of formal definitions. Although there are no formal definitions, their existence are known by human performers [9]; however, they are for some reason, e.g., due to previously unknown set of activities, being considered not valuable enough, etc., not formalized. As the main driver of the informal processes are human-decisions, they are quite volatile in their nature. Performers use IT resources to carry out these informal processes. During enactment of the informal processes, data can be created or consumed. An execution of a traditional business process might cause an informal process execution or in an informal process, formal process might be executed. Despite the fact that the informal processes are not formally documented, there might exist recurring activities and best-practices in these processes, which might be re-used. This work introduces a new concept of supporting these informal processes by using the available knowledge in these processes.

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

The main contributions of this paper can be listed as:

- An overview of requirements for supporting enactment of informal processes (Sect. 2)
- Introduction of the concepts of *Informal Process Support Model* (Sect. 3), *Informal Process Essentials* (Sect. 3.1) and *Informal Process Recommendations* (Sect. 3.2)
- A discussion on the related work (Sect. 4)

2 Requirements for Supporting Enactment of Informal Processes

According to Leymann and Roller [6], business processes have three dimensions, i.e., business logic, IT infrastructure, organization and we will analyze our requirements under these categories. Next, we will explain a simple motivating informal process scenario to describe requirements easier.

In an enterprise, a product team receives a new product feature request and makes an assessments of their available resources, i.e., human and IT resources, to satisfy corresponding requests. After assessment, a new expert is recruited, and he uses the software tools as the other members of the product team do to participate in this collaborative work. To satisfy the new feature request, the product team installs a new software tool, which will be used by all team members.

In the following section, we present the first set of requirements based on the dimensions of business processes, which came up in discussions with the scientific community. A proper justification is out of scope of this paper and follows in future work.

2.1 Business Logic

Business logic refers to the activities that need to be done to execute the corresponding process. In case of informal processes, business logic is quite different from standard business processes because there are no predefined steps and new steps can emerge in each execution. In the following paragraphs, we have the requirements for supporting enactment of informal process regarding business logic dimension.

Means of Providing Core Elements of an Informal Process **(R1)***:* Providing core elements of an informal process involves describing core elements, i.e., performers, IT tools, data and the means of making these resources ready, e.g., textual descriptions of how to make these ready. By satisfying this requirement, we define the main performers, tools and data to carry out corresponding informal process. In our motivating scenario, the core elements are the product team, their tools and the data used during product development.

Means of Supporting Performers without Constraining the Flexibility **(R2)**: Performers need to be guided without any constraints on the execution of informal

processes. By this way, we provide means of supporting the performers without dictating any activities. An example of this could be providing the related data to the new team member in our motivating scenario.

Means of Exploiting Existing Implicit Knowledge **(R3)***:* Previous executions of an informal process would contain resourceful information and usage of this information in the further executions would be needed. During satisfaction of a new product feature request, users could be guided with possible further actions.

2.2 IT Resources

Performers depend on IT resources for various reasons, e.g., finding some information, creating some artifacts, collaborating with each other, etc., to accomplish informal processes. In this section, we present the requirements for supporting enactment of informal processes regarding IT resources aspects.

Means of Inclusion of IT Resources **(R4)***:* IT resources influence outcome of the informal process and we need a means of associating them with informal processes. By associating the IT resources, we provide a common collaboration infrastructure where interoperability increases. In our motivating scenario, all the tools that are used by the product team for the feature request.

Means of Representing Relationships of Performers and the Tools of Inflexible Process **(R5)***:* Each role might have different kind of relationships with the software tools that they are using and we need a means of representing these, e.g., a regular user vs. an admin. By this way we can express different relationships of IT resources and the performers. In our motivating scenario, the access rights to the data of a temporarily recruited expert would be different than the other team members.

Means of Representing Informal Process Specific IT Resources **(R6)***:* Each informal process might have different set of software tools and we need a means of isolating informal processes from each other. By this way, each informal process has an isolated execution context. In our example the informal process of adding a new feature has its own set of tools which are shared by the performers.

Means of Changing the Set of Tools During Execution **(R7)***:* Performers might need additional tools and remove old ones as they desire. We need a means of changing the list of tools after initialization. In our motivating scenario, we have an additional tool for the new feature. This new tool is added to the informal process context and used by all the performers.

2.3 Organization

Organizational aspects of an informal process is important because they have a direct effect on the outcome of the informal process. Next, we present some requirements for supporting enactments of informal processes regarding organizational aspects.

Means of Representation of Human Performers and Their Relationships **(R8)***:* The performers of an informal process have some roles, skills and certain

relationships among each other which would influence the outcome. We need a means of associating these performers and relationships. As a result, we abstract our actual set of performers and they can be replaced with an equivalent set of performers in the next enactment. In our motivating, we would have a team with certain relationships among each other, e.g., manages relationship vs. recruits relationship.

Means of Addition and Removal of Performers During Execution **(R9)***:* In case of a lack of an expertise, new experts might be recruited and informal processes might be associated with new performers. We need a means of adding and removing performers from informal processes during the enactment of the business process. As a result, we do not limit the list of performers to problem. In our example, the product team recruits a new expert based on their needs.

Means of Providing Context Switching for a Performer **(R10)***:* A performer might be a performer in an another informal process and means of context switching is needed. As result, performers can be more productive and use the related tools and data for an informal process. In our motivating scenario, the recruited expert participates in more than one informal process and each of them has their own set of tools and data.

3 Informal Process Support Model

To satisfy these requirements, we introduce the concept of *Informal Process Support Model (IPSM)* (Fig. 1). The concept contains additional new concepts, i.e., *Informal Process Essentials (IPE)* and *Informal Process Recommendations (IPR)*. The concept of *IPE* follows a declarative approach by stating *what* the problem is and bringing together the core solution elements for the problem. *IPR* not only supports *IPE* with some improvement recommendations but also it recommends some action steps based on the past enactments of the corresponding informal process. In the following sections, we will detail these two main concepts.

3.1 Informal Process Essentials

*IPE*s describe not only the building blocks of informal processes, i.e., performers, data and software tools, but also they describe how to make the core elements ready for the enactment of the informal process, i.e., resource providers. The model provides the necessary concepts and relations for modeling core elements of an informal process. As a result, we can initialize for the enactment of an informal process and we meet the requirement **R1**.

*IPE*s include description of software tools that are used in the enactment of corresponding informal process and each informal process model can have its isolated set of tools. As a result each *IPE* is associated with informal process specific tools and data, which satisfy the requirements **R4** and **R6**. *IPE*s enable association of IT resources independent from the status of the corresponding IPE, e.g., running, suspended, etc., of the corresponding informal process, which

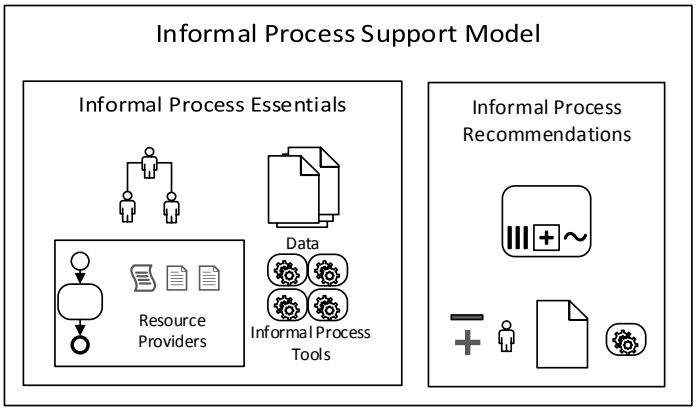


Figure 1. An abstract view on the *Informal Process Support Model*

satisfies the requirement **R7**. In *IPE*, with the relationships provided by IPE, custom relationships are possible and one can associate IT resources, i.e., data and software tools with human performers. As a result **R5** is satisfied.

In *IPE* both performers and their relationships can be defined. Performers are abstract description which are made concrete during initialization or update of an *IPE*. They have interrelations among each other. By providing means of defining a group of performers and their relationships, we satisfy the requirement **R8**. Addition and removal of performers do not depend on the status of corresponding informal process, which results in satisfaction of **R9**. For each informal process that a performer participating in, there are performer specific views based on the described software data and permissions in an *IPE*. These performer specific views provide an isolated view for the performers and a means of easy context-switching. By providing an isolated view, we satisfy **R10**.

Initially, *IPE*s are created by performers who have knowledge about the corresponding informal process and they are assumed to contain necessary set of elements to conclude a corresponding informal process. However, as the needs change the instances of the concepts can be updated at any time, e.g., adding new performers, tools, etc.

For realization of *IPE*, we need to model our performers, software tools and the related data. Moreover, we need a means of making these resources ready. Considering essential characteristics of cloud computing [8], cloud computing would be a good choice for automated provisioning of IT resources. Topology and Orchestration Specification for Cloud Applications (TOSCA) [3] provides means of modeling cloud IT infrastructures and data associated with the corresponding applications. It has a corresponding run-time container (aka. TOSCA container) [2] and an open source editor Winery [5]. In TOSCA models, one can define the topology of an application and how the corresponding application components can be provisioned in the form of business processes, e.g., using BPEL [11] or BPMN [12] or scripts. In our use case, topology templates could be used to

represent human performers, IT infrastructure and data concepts of an *IPE* and to describe how these resources are provisioned we could use corresponding deployment plans. The suitability of TOSCA for modeling *IPE*s needs to be further investigated and left as a future work.

3.2 Informal Process Recommendations

*IPR*s contain the tips which are gathered from previous executions. They guide the performers during modeling time of *IPSM*s and during run-time of an informal process. Considering that they are just "recommendations", we do not constrain the flexible execution of an informal process. However, we still make some recommendations based on the information collected and by doing so we satisfy **R2**. These recommendations are collected during the enactment of an informal process so they are based on the implicit knowledge of the performers. As a result, we use the best practices that the performers bring to the enactment of the informal process; therefor, we satisfy the **R3**. During realization of the concept of *IPR*, we will exploit the information created in the IT infrastructure where the informal process takes place.

4 Related Work and Discussion

BPEL4People [4] is an extension of BPEL to support people in business processes. However, it is not possible to change the model, after it is initialized, i.e., we have constant staff query and after assignment, it cannot be changed. The work of Shall et al. [14] introduces a framework for Human-provided Services, these services provide a unified interface for web-services and human services. By use of this framework people can publish services based on their skills. These can be used as a complementary to our approach during finding the corresponding resources; however, not an alternative.

Liptchinsky et al. [7] define a modeling framework for collaborations. An extended version of UML state diagram is used to represent collaboration artifacts, their relationships with performers and relationships of performers. They include additional transition concepts to design the flow of a collaboration process. The models do not include software tools (R1, R4, R5, R6, R7, R9 are not satisfied). By adding some states and some conditions, we limit the flexibility of the process (R2, R3 are not satisfied).

In the work of Papageorgiou et al. [13], based on some pattern definitions, users are guided through a collaboration process. Events are analyzed and they are mapped to some collaboration patterns and users are guided based on these patterns. Inclusion of software tools and some means of making the resources is not explained in the work (R1, R4, R5, R6, R7 are not satisfied). Moreover the authors do not mention addition and removal of performers during execution (R9 is not satisfied).

Activity-centric computing [1, 10] provides a platform where users can create activities, associate people, resources and to-do lists with these activities. Users

collaborate using some applications integrated to activity-based components. From an instance of an activity, an activity-pattern can be extracted and can be used for another similar case. In the concept of activity-centric, no concept of resource providers have been proposed, the relationships between software tools and performers have not been mentioned and relationships of performers cannot be represented (R1, R5 and R8 are not satisfied).

By our introduced concepts, we preserve the flexibility of informal process enactments. *IPE* provides the performers, tools and data for the enactment of an informal process and it is supported by the *IPR*. This model suits well because in case of informal processes the business logic not modeled beforehand and with this model, we do not enforce modeling it beforehand. After establishing an initial support model, we analyze the on-going collaborations in the context of corresponding informal process. Hereafter, we analyze the collaborations and we present the findings as recommendations to the performers.

5 Conclusions and Outlook

In this work, requirements for supporting enactment of informal processes have been introduced and with some new concepts, these requirements have been fulfilled. The concept of *IPSM* has the purpose of providing necessary elements for the conclusion of an informal process and assisting performers and *IPSM* designers. *IPE* is the concept which contains the descriptions of core elements of an informal process and *IPR* refers to the tips and recommendations to provide an easy execution of an informal process.

As the next step, requirements will be justified, the introduced concept of *IPE* will be detailed and a corresponding prototypical implementation will be provided. Thereafter, on top of the established concept of *IPE*, we will detail the concept of *IPR*.

Acknowledgments This work is supported by Graduate School of Excelence in Advanced Manufacturing[1] (GSaME).

References

1. Bailey, J., Kandogan, E., Haber, E., Maglio, P.P.: Activity-based management of IT service delivery. In: CHIMIT 2007. CHIMIT '07, ACM, New York, NY, USA (2007)
2. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In: ICSOC'13. LNCS, vol. 8274, pp. 692–695. Springer Berlin Heidelberg (Dec 2013)
3. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: chap. TOSCA: Portable Automated Deployment and Management of Cloud Applications, pp. 527–549. Springer, New York (Jan 2014)

[1] `http://www.gsame.uni-stuttgart.de/`

4. Ings, D., Clément, L., König, D., Mehta, V., Mueller, R., Rangaswamy, R., Rowley, M., Trickovic, I.: WS-BPEL extension for people (BPEL4People) specification version 1.1. OASIS Committee Specification (Aug 2010)
5. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – Modeling Tool for TOSCA-based Cloud Applications. In: ICSOC'13. LNCS, vol. 8274, pp. 700–704. Springer Berlin Heidelberg (Dec 2013)
6. Leymann, F., Roller, D.: Production Work Flow: Concepts and Techniques. Prentice Hall PTR (2000)
7. Liptchinsky, V., Khazankin, R., Truong, H.L., Dustdar, S.: A novel approach to modeling context-aware and social collaboration processes. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE, Lecture Notes in Computer Science, vol. 7328, pp. 565–580. Springer Berlin Heidelberg (2012)
8. Mell, P., Grance, T.: The NIST definition of cloud computing (draft). NIST special publication 800(145), 7 (2011)
9. Moody, P., Gruen, D., Muller, M., Tang, J., Moran, T.: Business activity patterns: A new model for collaborative business applications. IBM Systems Journal 45(4), 683–694 (2006)
10. Moran, T.P., Cozzi, A., Farrell, S.P.: Unified activity management: supporting people in e-business. Commun. ACM 48(12), 67–70 (Dec 2005)
11. OASIS: Web Services Business Process Execution Language Version 2.0 – OASIS Standard (2007)
12. Object Management Group: Business process model and notation version (BPMN) version 2.0 specification. Tech. rep., Object Management Group (OMG) (Mar 2011)
13. Papageorgiou, N., Verginadis, Y., Apostolou, D., Mentzas, G.: Event-driven adaptive collaboration using semantically-enriched patterns. Expert Systems with Applications 38(12), 15409–15424 (2011)
14. Schall, D., Truong, H., Dustdar, S.: The human-provided services framework. In: E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on. pp. 149–156 (Jul 2008)

Towards Standard Conformant BPEL Engines: The Case of Static Analysis

Christian R. Preißinger, Simon Harrer, Stephan J. A. Schuberth, David Bimamisa, and Guido Wirtz

Distributed Systems Group, University of Bamberg, Germany
{simon.harrer,guido.wirtz}@uni-bamberg.de
{david-chaka-basugi.bimamisa,christian-roland.preissinger,stephan-johannes-albert.schuberth}@stud.uni-bamberg.de

Abstract. The errors in BPEL processes that are only detected at runtime are expensive to fix. Several modelers and process engines for BPEL exist, and the standard defines basic static analysis (SA) rules as a detection mechanism for invalid processes, but the actual conformance of BPEL modelers and engines regarding these rules is unknown. We propose to develop test cases to evaluate the conformance of BPEL modelers and engines regarding static analysis. The evaluation results enable decision makers to identify and use the most conformant engine and modeler that detect errors before runtime and therefore reduce costs.

Keywords: SOA, BPEL, static analysis, conformance testing

1 Motivation

BPEL, a standard [9] by OASIS, defines a graph and block structured process language (see [6]), corresponding execution semantics, and 94 basic static analysis (SA) rules[1]. "The purpose of [these rules] is to detect any undefined semantics or invalid semantics within a process definition that was not detected during the schema validation against the XSD" [9, p. 194]. These rules seem rather simple, but are nevertheless as important for executing BPEL processes as static type checking is for executing Java applications. Consequently, one would expect the IDEs (BPEL modelers) and the runtime (BPEL engines) to detect any violations of these rules. Especially, as the BPEL specification requires a fully standard conformant engine to implement all static analysis rules [9, p. 13].

Each static analysis rule defines constraints for at least one BPEL element, e.g., rule #47 enforces, among other conditions, that any received non-empty message must be stored in variables. For example, consider a developer implements a simple BPEL process[2] with two variables (input and output variable) which awaits a message (`onMessage`) which instantiates the process, copies the contents of the

[1] The rules are enumerated from 1 to 95, but 49 is missing, thus 94 rules exist.

[2] The process is available at `https://lspi.wiai.uni-bamberg.de/svn/betsy/sa47-test.zip` - Accessed 02/14/2014

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at `http://ceur-ws.org`

input variable into the output variable (`assign`) and returns the output variable (`reply`), but forgets to store the received message in the designated input variable, hence violating rule #47. Against expectations, the error is neither detected by the two widely used Open Source BPEL IDEs, *Eclipse BPEL Designer v1.0.3* and the *OpenESB IDE v2.3.1*, nor by the BPEL analysis tool *BPEL2oWFN*. Moreover, only the BPEL engines *Apache ODE 1.3.6* and *bpel-g v5.3* correctly reject the erroneous process whereas *OpenESB v2.3.1* and *Orchestra 4.9* falsely accept and deploy the process. Upon execution of the process, both engines show behavior which is hard to debug: *OpenESB* returns `null` with the error of a `NullPointerException` and *Orchestra* returns a timeout with no error trace. In this particular case, none of the modelers or analysis tools and only two out of four engines were able to detect this basic error.

Thus, this preliminary evaluation raises the following research question: *What is the conformance to the static analysis rules of BPEL modelers and engines and why is this the case?*

2 Related Work

Static analysis regarding BPEL has been studied extensively in literature. Each considered approach was analyzed by investigating three aspects: the BPEL version, the number of test cases, and the amount of static analysis rules covered by the tests. The approaches [1–3, 11] focus on BPEL 1.1 whereas [7, 12, 13] focus on the latest specification BPEL 2.0. Because the rules were initially published in the BPEL 2.0 specification, the first four approaches could not specify any SA conformance tests. Nevertheless, Akehurst [1] and Ouyang et al. [11] provide 16 and 30 valid BPEL 1.1 processes as test cases, respectively. In addition, Ouyang et al. [11] presents two incomplete BPEL 1.1 processes detailing an unreachable activity and a conflicting `receive`, the latter would violate the rule #60 of BPEL 2.0 which requires the use of explicit `messageExchanges` in this case. Returning to the approaches using BPEL 2.0, only [7] provides test cases[3]. Each of these 56 test cases corresponds to a specific static analysis rule. Moreover, Lohmann presents the tool *BPEL2oWFN* which automatically detects violations of these 56 rules as a positive side effect during a transformation from BPEL to Petri Nets [8, p. 34]. Because of a different focus of [7], the 56 provided tests are not suitable to evaluate the conformance to the static analysis rules of BPEL engines. They do not include all error types of the covered 56 rules and are abstract (no WSDL interface, incomplete process definition).

Whereas the discussed approaches check the standard conformance of BPEL processes, none of them evaluates the standard conformance of BPEL modelers or engines. Harrer et al. [4, 5] did focus on BPEL engines with their automated testing tool *betsy*, but solely evaluated standard conformance using approx. 130 valid processes. Thus, standard conformance regarding the static analysis rules of BPEL engines remains untested [4, p. 7], as is the case for BPEL modelers.

[3] The test cases are available as part of the source code of the `BPEL2oWFN` tool at `http://www.gnu.org/software/bpel2owfn/download.html` - Accessed 01/22/2014

3 Research Outline

To answer the research question, we aim to a) create test cases ($\xrightarrow{\text{derive}}$ in Fig. 1) for the 94 static analysis rules and b) use these test cases to analyze static analysis conformance of BPEL modelers and engines ($\xrightarrow{\text{evaluate}}$ in Fig. 1) with *betsy*, i.e., determining the degree of static analysis conformance.

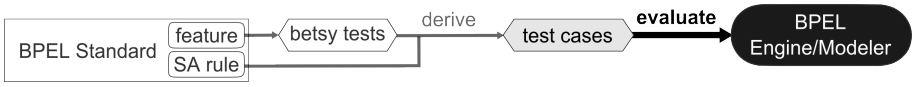


Fig. 1. Big Picture of our Approach

The derivation of the test cases for each static analysis rule is subdivided into six steps. First, the related elements and attributes are extracted from the textual representation of the rule and partitioned into groups. Second, the elements and attributes are permutated into a list of possible combinations. Third, we identify valid, invalid, and meaningless combinations according to the standard restrictions. Fourth, we calculate the distance from each invalid combination to every valid combination and pair up the least distant combinations. Our distance metric is the amount added and removed elements and attributes. Fifth, to create the test case we select the most feature-poor process from a test set of valid processes (e.g. the *betsy* test set) that fits the valid combination. Sixth, we mutate the valid process to an invalid one corresponding to the invalid combination. This approach increases the quality of the tests as it ensures that the erroneous process solely violates a single error condition. Especially the first and the third step are hard to automate as interpreting prose is nontrivial. Thus, four steps are automated whereas the other two are done manually. Following our running example, we applied the first step of our proposed procedure onto rule #47 and determined the formalization of influencing factors in the listing below. Step two to five are not shown. Regarding step six, the test case described in Sect. 1 implements the error condition represented by the permutation marked as bold.

{empty message, **non-empty message**} ×
{invoke, receive, reply, **onMessage**, onEvent}×
{**incoming**, outgoing}×
{variable assignment, **no variable assignment**}×
{part assignment, **no part assignment**}

An engine passes such a test case if the process is rejected during deployment. But there may be false positives, i.e., the process is rejected by an engine because of an unsupported feature or an internal error. To prevent misleading results, we propose to take the betsy conformance evaluation into account. betsy reveals [4, p. 6] that full standard conformance is far from given by detailing which BPEL feature is supported by which engine. To avoid false positives during the evaluation of a single error type, we suggest to use a pair of BPEL processes,

a fully functional and an erroneous one. We assume that if the engine rejects the valid process, it is not able to detect this error type. But this assumption introduces false negatives, as the engine may reject the erroneous process by static analysis and the valid one by missing feature support. To counter this, we propose to evaluate the log files for any hints on why the deployment failed. The evaluation of the BPEL modelers is analogous.

The evaluation with test pairs of valid and invalid tests reveal the quality of the error detection, making the engines and modelers comparable in this regard. As the degree of error detection has an impact on the development costs, this metric may be leveraged for buying decisions. In addition, it can also be used for improving the products and act as a regression test suite.

The requirements to use our approach are 1) a process language standard defining rules for valid processes and 2) a feature complete set of valid processes. We use the process language BPEL in this case study, but the approach is also applicable for other process languages, e.g., for the Business Process Modeling and Notation (BPMN) [10].

References

1. Akehurst, D.H.: Validating BPEL Specifications using OCL. Tech. Rep. 15-04, University of Kent, Computing Laboratory (August 2004)
2. Fisteus, J.A., Fernández, L.S., Kloos, C.D.: Formal verification of BPEL4WS business collaborations. In: EC-Web LNCS 3182, pp. 76–85. Springer (2004)
3. Foster, H., Uchitel, S., Magee, J., Kramer, J.: LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography. In: ACM ICSE (2006)
4. Harrer, S., Lenhard, J., Wirtz, G.: BPEL Conformance in Open Source Engines. In: IEEE SOCA (2012)
5. Harrer, S., Lenhard, J., Wirtz, G.: Open Source versus Proprietary Software in Service-Orientation: The Case of BPEL Engines. In: ICSOC. LCNS, vol. 8274, pp. 99–113. Springer Berlin Heidelberg, Berlin, Germany (2013)
6. Kopp, O., Martin, D., Wutke, D., Leymann, F.: The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. Enterprise Modelling and Information Systems 4(1), 3–13 (June 2009)
7. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: LNCS, 4th WS-FM (2007)
8. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0 and its compiler BPEL2oWFN. Tech. rep., 212, HU Berlin (August 2007)
9. OASIS: Web Services Business Process Execution Language (April 2007), v2.0
10. OMG: Business Process Model and Notation (January 2011), v2.0
11. Ouyang, C., Verbeek, E., van der Aalst, W., Breutel, S., Dumas, M., ter Hofstede, A.: WofBPEL: A Tool for Automated Analysis of BPEL Processes. In: ICSOC (2005)
12. Yang, X., Huang, J., Gong, Y.: Defect Analysis Respecting Dead Path Elimination in BPEL Process. In: IEEE APSCC (2010)
13. Ye, K., Huang, J., Gong, Y., Yang, X.: A Static Analysis Method of WSDLRelated Defect Pattern in BPEL. In: IEEE ICCET (2010)

ViePEP – A BPMS for Elastic Processes

Philipp Hoenisch

Distributed Systems Group, Vienna University of Technology, Austria
p.hoenisch@infosys.tuwien.ac.at

Abstract. In today's IT industry resource-intensive tasks are playing an increasing role in business processes. By the emergence of Cloud computing it is nowadays possible to deploy such tasks onto computing resources leased in an on-demand fashion from Cloud providers. This enabled the realization of so-called Elastic Processes (EPs). These are able to dynamically adjust their used resources in order to meet varying workloads. Till now, traditional Business Process Management Systems (BPMSs) do not consider the needs of Elastic Processes such as monitoring the current system load, reasoning about optimally utilized resources, in order to ensure given Quality of Service constraints while executing required actions such as starting, stopping servers or moving services from one server to an other. This paper focuses on our current work on ViePEP, a research BPMS for the Cloud capable of handling the aforementioned requirements of EPs.

1 Introduction

Business Process Management is a multifaceted approach which covers the organizational, management and technical aspects of business processes. Further, it "includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes" [1]. In recent years, a specific subtopic of business process management gained more attention in many industries: the automatic processing of business processes also known as workflows (excluding the involvement of human services). In many cases, software services are composed to a workflow in order to realize a specific functionality. Therefore, by its nature, the individual (software) services in such a composition differ in terms of required computing resources (such as CPU, RAM, bandwidth, ...), priority and execution order. In order to realize and process such a workflow, different techniques, concepts, methodologies and frameworks from the field of computer science are required.

Such workflows are becoming more and more relevant in business processes in several different industries. Examples are coming from the finance industry, managing of smart grids or from the energy domain. In the latter one, data from a very large extend of sensors have to be gathered, processed and analyzed in almost real time. Further, this data has to be stored in order to be retrievable to a later moment for the generation of reports or statistical analysis.

It is a common service provider problem that acquired resources are hardly fully utilized, which is not very cost efficient. While this enables the provision of

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

a high quality of service, it ends up in unwanted waste of resources. In contrast, if too many requests are forwarded to a particular Virtual Machine (VM) it may crash or the services being executed may produce faulty results. As this example is very specific for computer engineering, it is a common problem in economy. In computational processes in the context of Cloud computing, this can be described as the problem of *Elastic Processes* (EPs). EPs are "precisely defining the various facets of elasticity that capture process dynamics in cloud computing [...]. The main properties for modeling EPs' economic and physical dynamics are *resource elasticity*, *cost elasticity*, and *quality elasticity*" [7]. While EPs are a complex concept, the problem around it can be stated as: Finding the correct relation between Resources, Costs and Service Quality, or in other words: Acquire as little resources as required in order to ensure the best possible quality of service while only paying the least required amount.

Therefore, a technology is needed which is able react to a dynamic change of needed computing resources, while still ensuring the faultless business process execution. This means, this kind of technology has to be able to provide additional resources when needed, such that, the business process execution will not wait, stuck or even crash at a critical moment. For that reason, research scientists from the field of Business Process Management and software engineering have put a remarkable focus on the solution of such a problem in recent years.

In this paper, we present the ongoing research on Elastic Processes in the Cloud. More precisely, we present our extensive work on *ViePEP* – the *Vienna Platform for Elastic Processes*. ViePEP is a research-driven BPMS for the Cloud, capable of cost-effective workflow processing while monitoring their underlying service executions in order to provide a certain level of Quality-of-Service (QoS) and ensure no Service Level Agreement (SLA) violations.

The remainder of this paper is organized as follows: After a brief introduction of ViePEP's architecture including its functionality (Sect. 2) we will give some information about our current research on workflow scheduling (Sect. 3.1) and resource optimization (Sect. 3.2). Sect. 4 will give an overview of the related work and Sect. 5 will conclude this paper and give a short outlook on our future work.

2 The Vienna Platform for Elastic Processes

In this section we want to introduce ViePEP. In general, ViePEP can be seen as a broker middleware which accepts workflow requests by a customer (Client in Fig. 1) and takes care of its execution. By the upcoming of Cloud computing and the new paradigm of Software-as-a-Service (SaaS) [4], many business processes are already SaaS-enabled, which means, they can be deployed independently and reusable in the Cloud. ViePEP takes care of the hosting and managing of the software services and maps the clients workflow requests in order to execute them. In addition, ViePEP considers the Service Level Agreements, which may be defined by clients. In order to accept hundreds of workflow requests simultaneously while still being able to ensure the given SLAs and being as

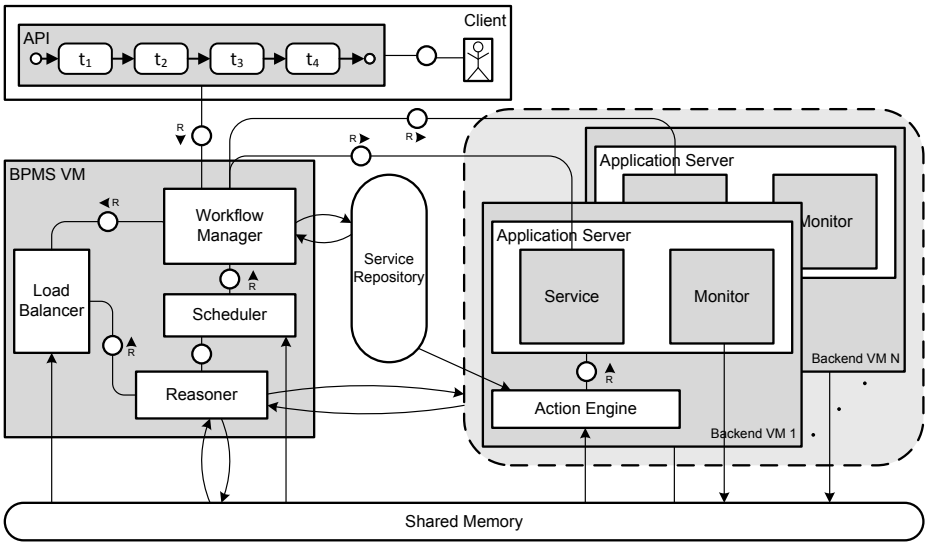


Fig. 1. ViePEP – Architecture

cost-efficient as possible, ViePEP was designed according to the MAPE-K cycle (*Monitor*, *Analyze*, *Plan* and *Execute*) which is used for autonomic computing[12].

As shown in Fig. 1, ViePEP has five top level entities: First, the *Client* models service-based workflows and can optionally define Service Level Agreements. Clients may request additional workflows consecutively or even many simultaneously. In addition, ViePEP is able to serve several different clients in parallel.

Second, the *BPMS VM* offers the core functionality of ViePEP. It is responsible of accepting new workflow requests (*Workflow Manager*) and stores them for a later or immediate execution. The exact execution time is computed by the *Scheduler*, which creates a schedule plan according the given deadlines defined in the given SLAs within the workflow requests. A first version of this scheduling plan is forwarded to the *Reasoner* which computes the amount of required resources. This can be done by reasoning on historical data from the Shared Memory. As this is the core functionality it will be further discussed in Sect. 3. Thus acquired resources are used equally, the single service invocations are balanced and distributed to the single service instances running on different VMs (Backend VM). Beside of the workflow executions, the Workflow Manager also measures the execution time of single service invocations, which is a prerequisite to detect possible deviations from the expected QoS attributes. By doing so, it is able to issue corresponding countermeasures if required.

Third, the *Backend VM* hosts an *Application Server* on which a particular service instance is deployed. In order to monitor the services' QoS, a *Monitoring* component is deployed. It measures the VM's CPU and RAM load and stores this information in the Shared Memory. The *Action Engine* is able to perform

actions issued by the Reasoner such as *deploy*, *undeploy* a particular service, or *move* a running service to another Backend VM.

Fourth, both, the *Shared Memory* and *Service Repository* are helper components and their functionalities are simple. The latter hosts all available services in form of deployable Web application ARchive (WAR) files. The Shared Memory is used to store the monitored data from each single Backend VM and share it with the BPMS VM. For a more detailed description about ViePEP please be refereed to [8,9].

3 Scheduling, Reasoning and Optimization

As ViePEP is a fully functional BPMS for the Cloud it takes care of workflow scheduling (Sect. 3.1) and its actual workflow execution. However, in contrast to common BPMSs, ViePEP is considering the future workflow executions and reasons in order to achieve a cost-effective optimized system (Sect. 3.2). For that, we will discuss in this section our ongoing work on the core functionality of ViePEP: the reasoning about current and future workflow execution including the computation of the resource demand and how ViePEP achieves a resource optimized system landscape.

3.1 Scheduling

The core functionality of a common BPMS is to process workflows. In order to know when a particular workflow execution should be started, several different procedures have been established. In many cases the incoming workflows are first ordered according their priorities before being processed. This allows the processing of workflow requests with a higher priority before workflows with a lower priority. These different techniques have already been discussed by many researchers and are not focus of this work [17]. ViePEP is making use of a priority-based scheduling approach, i. e. workflows with a higher priority are processed before workflows with a lower priority. Priorities are calculated based on the deadline defined in the given SLAs of the workflow requests. If two or more clients have defined the same deadline for different workflows, ViePEP will serve them according a first-come first-serve manner. However, as ViePEP is able to process several workflows simultaneously while considering each given SLA, the clients will not notice any delay.

Clients can issue a workflow request and define a specific deadline, i. e. a point of time defining when the execution has to be ended. This can be defined either for the whole workflow or for a particular single step in a workflow. ViePEP's task is to process the workflow while ensuring this deadline. Since ViePEP is a BPMS serving several hundred or even thousand clients in parallel, the workflow scheduling is a complex task. Hoenisch et al., [8] describes the latest implemented scheduling algorithm. It splits up a workflow into its single steps and assigns them to a particular time slot. Each time slot is exactly as long as the single service invocation lasts. Service invocations of the same type, i. e. the same kind

of software service has to be invoked, can be combined within the same time slot in order to make fully use of the acquired resources.

This scheduling is a straight forward task for sequential workflows (which are the only one supported in ViePEP at the moment). However, it gets a much more complicated challenge if the workflow is more realistic, e. g. if it involves branches such as XORs, ANDs or loops. While ANDs are quite easy to implement, i. e. both branches have to be considered, XORs are much more complicated. In the latter one, a BPMS, considering XORs has to deal with probabilities. This means, it has to calculate how high is the chance that a workflow follows either the one path or the other one (but not both). This can either be done static, e. g. the probability that a workflow follows the one path is always a fix value and the other direction is always 1 minus that value, or dynamically. Of course, in a real world scenario, those values are not static, but may change dynamically, e. g. they depend on the output or input of previous steps. Therefore, a "smart" BPMS has to be able to learn from historical executions and predict the probabilities. While the scheduling itself might not be the biggest challenge, as already a lot of research has happened in this field, the combination with computing the demand of resources is much more complicated.

3.2 Reasoning & Optimization

As ViePEP is a smart BPMS it tries to consider all of the three properties of EPs equally. This means, the acquired resources are fully utilized in order to be as cost-efficient as possible. However, in the current version, the quality of a service is hardly defined by the services' output, rather than ensuring the given SLAs, i. e. ensuring that a workflow is processed in time.

Resource Prediction
As mentioned before, ViePEP tries to utilize the acquired resources as efficiently as possible. This means, the Reasoner computes the required amount of resources from historical data. In the current version, ViePEP makes use of Ordinary Least Squares (OLS) Linear Regression. At the moment, the provided services are only CPU intensive. Therefore, a high CPU load would influence a hosted service the most. Therefore, OLS is a perfect choice for the current scenarios as it is limited to two variables (2-dimensional optimization). While this is only the case in our selected services, in the case of image processing, the limiting factor may be the internal memory or RAM. For that reason, we are working on a multi-dimensional resource prediction mechanism considering several QoS aspects of a service. As in real-world scenarios, service invocations do not produce a linear resource consumption, and may last several minutes or even hours, a linear regression is not applicable anymore. Therefore, we propose to approach this problem from the other way around and make use of online reasoning approaches (e. g. *Kalman Filters*) in order to compute how many service invocations are possible on a particular resource and to predict the future demand of resources. In general, a Kalman Filter aims at providing the means of a mathematical equation to estimate a state of a process or a stream of updated data. In addition, a Kalman

filter makes use of historical and life data and is able to predict a future state even if the precise nature of the system is not known. Therefore, by "feeding" a Kalman Filter with monitored data, e. g. such as how many invocations happened in parallel, producing a certain load in CPU and used a particular amount of RAM, it is possible to compute how many invocations the monitored VM is able to handle in the future.

Resource Allocation
The result of the resource prediction (see Sect. 3.1) is a detailed plan of which service invocation is assigned to which VM and if additional VMs are required or if unneeded once can be released. The execution of this task is simple software engineering.

However, in many cases, companies manage an own private Cloud. Which means, neglecting the energy costs, these are free resources and ready to use. Therefore, the Reasoner should consider allocating private resources first until the demand is reached. However, it may be the case, that not enough resources are available in the private Cloud. Therefore, additional resources can be bought from an external Cloud provider (public Cloud). The result is a so called Hybrid Cloud. As the resource allocation on its own is not such a complicated task, the Reasoner has to consider the different pricing schemes of the public Cloud. Amazon's EC2 for example charges their customers on an hourly model. This means, it is not economic to acquire such a resource for just 20 minutes. Therefore a rescheduling might be necessary. In addition, several Cloud infrastructure providers offer different kind of VM types having a different amount of computing resources such as a multi-core CPU or more RAM and cost differently.

4 Related Work

To the best of our knowledge, so far, surprisingly little effort has been investigated into the field of elastic processes in the sense of dynamic resource allocation and elastic process execution [7]. Nevertheless, there is some related work which remains to be mentioned from the fields of Grid computing and Cloud computing.

In both cases, scalability and cost-effective allocation of single tasks and services have been the only focus by many researchers. Most research efforts are focusing on minimizing the costs for the consumers (clients) while taking into account a maximum allowed execution time or other QoS attributes [5,14]. However, in later research, new approaches also consider SLA enforcement including the consideration of penalty costs. This lead a completely different approach of resource management in a Clout environment [3,6].

In contrast to that, more recent research efforts are focusing on the infrastructure perspective, i. e. a higher resource utilization [13,16] or maximizing the Cloud provider's profit [15]. While most of the time, only rule-based thresholds are applied to identify whether a new resources are required or unneeded can be freed, Li at al. [16] makes use of automated machine learning to scale applications up or down. However, all these approaches lack of the consideration of the

process perspective but focus on an ad-hoc allocation of Cloud-based resources for single services. Only a few research already considered a process perspective in regard for Scientific Workflows [10,11]. Although Business Processes and Scientific Workflows share several similarities, they also differ vastly in the sense of timeliness. The latter one can be run sometimes during the night, ensuring only the availability of the result in the morning, in Business Processes, the requests has to be processed in almost real time.

Similar to our work, a workflow model, i. e. workflows are composed from single software services which can be deployed in the Cloud, has been considered by Wei and Blake [18] and Bessai et al. [2]. Nevertheless, only one workflow is considered simultaneously which is one of the main focuses of ViePEP.

5 Conclusion

In this paper we have presented our current state and work on ViePEP – the Vienna Platform for Elastic Processes. ViePEP was already evaluated by simplified use cases in [9,8]. However, although we have shown that ViePEP is able to handle the presented use cases, optimize the acquired resources by rescheduling incoming workflows in order to be cost efficient as possible, ViePEP is still not yet fully supporting *Elastic Processes* [7], which is heavily focused by us in our future work. Therefore, we are extending ViePEP in order to support more realistic workflows including branches and loops. Further, it is planned to evaluate ViePEP on an hybrid Cloud environment involving Amazon's EC2, Windows Azure and others. In addition to that, an interested reader may have noticed that ViePEP and the Shared Memory may result in a bottleneck as well. While we already considered the latter one, and replaced the Shared Memory with a lightweight JMS Queue, the scalability of the BPMS VM is still part of our future work.

Acknowledgements. This work is partially supported by the Commission of the European Union within the SIMPLI-CITY FP7-ICT project (Grant agreement no. 318201).

References

1. van der Aalst, W.M.P., Hofstede, A.H.M.T., Weske, M.: Business Process Management: A Survey. In: International Conference on Business Process Management (BPM 2003). pp. 1–12. Springer, Berlin Heidelberg (2003)
2. Bessai, K., Youcef, S., Oulamara, A., Godart, C., Nurcan, S.: Resources allocation and scheduling approaches for business process applications in Cloud contexts. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings (CloudCom 2012). pp. 496–503. IEEE Computer Society, Washington, DC, USA (2012)
3. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In: 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010). Lecture Notes in Computer Science, vol. 6081, pp. 13–31. Springer, Berlin Heidelberg (2010)

4. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computing Systems 25(6), 599–616 (2009)
5. Cao, Q., Wei, Z.B., Gong, W.M.: An Optimized Algorithm for Task Scheduling Based on Activity Based Costing in Cloud Computing. In: 3rd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 2009). pp. 1–3. IEEE Computer Society, Washington, DC, USA (2009)
6. Cardellini, V., Casalicchio, E., Lo Presti, F., Silvestri, L.: SLA-aware Resource Management for Application Service Providers in the Cloud. In: First International Symposium on Network Cloud Computing and Applications (NCCA '11). pp. 20–27. IEEE Computer Society, Washington, DC, USA (2011)
7. Dustdar, S., Guo, Y., Satzger, B., Truong, H.L.: Principles of Elastic Processes. IEEE Internet Computing 15(5), 66–71 (2011)
8. Hoenisch, P., Schulte, S., Dustdar, S.: Workflow Scheduling and Resource Allocation for Cloud-based Execution of Elastic Processes. In: IEEE 6th International Conference on Service Oriented Computing and Applications (SOCA 2013). pp. 1–9. IEEE (2013)
9. Hoenisch, P., Schulte, S., Dustdar, S., Venugopal, S.: Self-Adaptive Resource Allocation for Elastic Process Execution. In: IEEE 6th International Conference on Cloud Computing (CLOUD 2013). pp. 220–227. IEEE (2013)
10. Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., Good, J.: On the Use of Cloud Computing for Scientific Workflows. In: IEEE Fourth International Conference on e-Science (eScience'08). pp. 640–645. IEEE Computer Society, Washington, DC, USA (2008)
11. Juve, G., Deelman, E.: Scientific Workflows and Clouds. ACM Crossroads 16(3), 14–18 (2010)
12. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer 36(1), 41–50 (2003)
13. Kertesz, A., Kecskemeti, G., Brandic, I.: An Interoperable and Self-adaptive Approach for SLA-based Service Virtualization in Heterogeneous Cloud Environments (forthcoming). Future Generation Computer Systems NN(NN), NN–NN (2012)
14. Lampe, U., Mayer, T., Hiemer, J., Schuller, D., Steinmetz, R.: Enabling Cost-Efficient Software Service Distribution in Infrastructure Clouds at Run Time. In: 4th IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2011). pp. 1–8. IEEE Computer Society, Washington, DC, USA (2011)
15. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B.: Profit-Driven Service Request Scheduling in Clouds. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010). pp. 15–24. IEEE Computer Society, Washington, DC, USA (2010)
16. Li, H., Venugopal, S.: Using Reinforcement Learning for Controlling an Elastic Web Application Hosting Platform. In: 8th International Conference on Autonomic Computing (ICAC 2011). pp. 205–208. ACM, New York, NY, USA (2011)
17. Schulte, S., Schuller, D., Hoenisch, P., Lampe, U., Dustdar, S., Steinmetz, R.: Cost-Driven Optimization of Cloud Resource Allocation for Elastic Processes. International Journal of Cloud Computing 1(2), 1–14 (2013)
18. Wei, Y., Blake, M.B.: Adaptive Service Workflow Configuration and Agent-Based Virtual Resource Management in the Cloud. In: 2013 IEEE International Conference on Cloud Engineering (IC2E 2013). pp. 279–284. IEEE Computer Society, Washington, DC, USA (2013)

Vinothek – A Self-Service Portal for TOSCA

Uwe Breitenbücher[1], Tobias Binz[1], Oliver Kopp[1,2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
[2] Institute for Parallel and Distributed Systems, University of Stuttgart, Germany
lastname@informatik.uni-stuttgart.de

Abstract The TOSCA standard provides a means to describe Cloud applications and their management in a portable way. TOSCA-based applications can be deployed on various standard-compliant TOSCA Runtimes. Vinothek is a Web-based Self-Service Portal that hides the technical details of TOSCA Runtimes and provides end users a simple graphical interface to provision Cloud applications on demand. This demonstration shows how Vinothek supports automated provisioning of applications and how it facilitates integrating TOSCA Runtimes.

Keywords: Cloud Applications; Self-Service Portal; TOSCA; Portability

1 Introduction

The *Topology and Orchestration Specification for Cloud Applications* (TOSCA [4]) is an OASIS standard for automating provisioning and management of Cloud-based applications in a portable and interoperable way. TOSCA provides a generic specification to model topologies that define the structure of Cloud applications. Topologies and all required software artifacts such as virtual machine images or scripts are stored in a package called *Cloud Service Archive* (CSAR). This archive is also standardized and enables vendors to distribute their applications to multiple Cloud providers based on a uniform and portable format.

To run TOSCA-based applications, a so called *TOSCA Runtime* is employed. TOSCA Runtimes consume CSARs, install them, and provide functionalities to provision and manage instances of the application. However, different TOSCA Runtimes provide different management APIs as these are not standardized by the specification: Some TOSCA Runtimes provide management functionalities out of the box, others require so called *Management Plans* which are contained in CSARs. Despite the goal of portability, TOSCA is currently lacking a common API specification to provision new application instances.

This demonstration tackles this issue by presenting the Self-Service Portal "Vinothek", which allows end users to provision new application instances through a simple, graphical user interface. Vinothek hides the technical details and differences of TOSCA Runtimes and provides a single unified interface to provision applications on various connected TOSCA Runtimes. Vinothek is based on Web technologies such as HTML5 and JavaScript and requires no additional software on client side. More details about TOSCA and TOSCA Runtimes are provided by the TOSCA Specification [4], the TOSCA Primer [5], and Binz et al. [2].

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

2 User Interaction

Vinothek provides a simple and easy accessible Web-based end user interface to provision new application instances on different TOSCA Runtimes. The portal consists of two main screens: The *Overview* page shown in Fig. 1 lists all applications installed in the TOSCA Runtimes connected to the Vinothek. The shown applications can be provisioned by the user. Therefore, the user selects one of the listed applications which leads to the *Application Details* page shown in Fig. 2. This page presents all details about the selected application and offers different *options* to configure the provisioning (shown on the bottom left in Fig. 2). The user provisions a new application instance by clicking the "Start Instance" button. If the provisioning requires additional user input such as payment information, a popup appears that enables filling in this information. After the provisioning is finished, the new application instance is opened in a new browser window. Depending on the type of the application, the user interface, a status page, or a remote desktop of the application instance is shown.

3 System Overview

The Vinothek is implemented following the Web-based client-server architecture shown in Fig. 3. The *Graphical User Interface (GUI)* is based on Java Server Pages and HTML5. It communicates via a *RESTful API* with the server that delegates calls to the *TOSCA Application Lifecycle Manager*, which is currently dealing with the provisioning of applications only. We plan to extend this component in the future to support management and termination of application

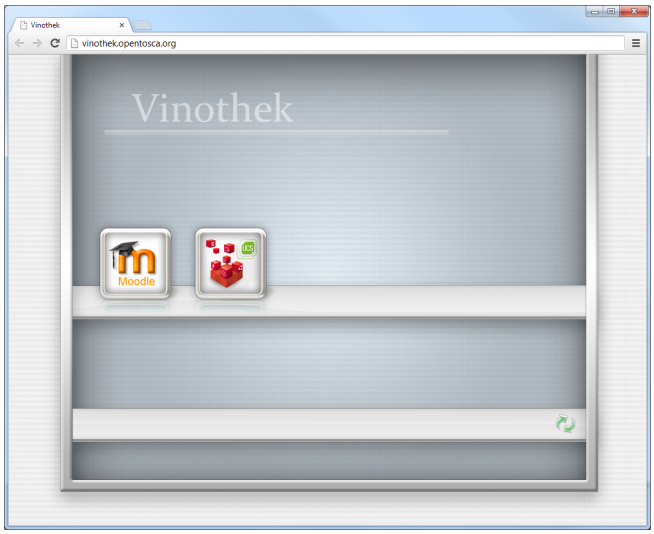


Figure 1. Vinothek Overview page showing all available applications

Figure 2. Vinothek Application Details page showing the selected Moodle application

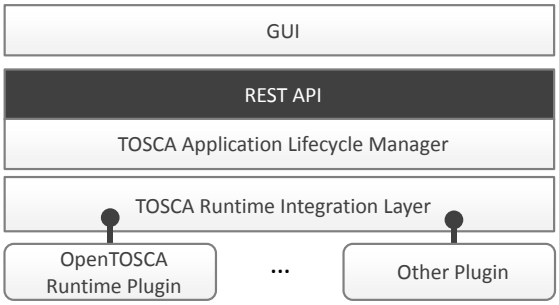


Figure 3. Vinothek System Overview

instances, too. Below this manager, the *TOSCA Runtime Integration Layer* provides mechanisms to plug-in TOSCA Runtimes. Plugins hook into the Vinothek Lifecycle Manager by implementing a certain interface provided by the integration layer and encapsulate all runtime-specific mechanisms to (i) provision a new application instance and (ii) to retrieve available applications that are installed as CSARs in the respective TOSCA Runtime. Thus, if a new application gets installed in a TOSCA Runtime that is connected to the Vinothek by a plugin, the new application is offered automatically by the Vinothek.

Depending on the API provided by the respective TOSCA Runtime, the implementations of plugins differ from each other. We implemented one plugin for the OpenTOSCA Runtime [1], which employs management plans implemented as workflows to provision and manage applications. The OpenTOSCA plugin connects, therefore, to (i) OpenTOSCA's workflow engine to provision new

application instances and to (ii) the RESTful OpenTOSCA Management API for retrieving installed applications.

As the TOSCA Specification does not define how to deal with self-service information, we extended the structure of CSARs by adding a "Meta-SelfService" folder. This folder contains a uniform XML-based application description including marketing information such as text, icons, and screenshots as well as a technical *Deployment Descriptor*. This Deployment Descriptor defines technical information required to provision the application on the respective runtime, i. e., required input parameters and runtime-specific information. When the provisioning of a new application instance is triggered, the Vinothek first requests all specified input parameters from the user via a popup. This information is passed to the plugin that uses these parameters and the technical information contained in the runtime-specific part of the Deployment Descriptor to start the provisioning of the application in the respective TOSCA Runtime. For example, the Deployment Descriptor of the school learning software "Moodle"[3] requires the initial username and password for the admin from the user. In addition, the Deployment Descriptor contains information required by OpenTOSCA to run the plans, e. g., it specifies that both Moodle database and business logic shall run in one single virtual machine. The "Meta-SelfService" folder itself may be created manually or by using the TOSCA modeling tool "Winery" [3].

4 Conclusion and Outlook

We presented the Self-Service Portal "Vinothek", which provides a simple graphical user interface for the provisioning of TOSCA-based applications. The tool also provides a means to integrate different TOSCA Runtimes transparently to end users and hides the technical details. A video of the demonstration is available at `http://demo.opentosca.org`. In the future, we plan to extend the Vinothek to support management functionalities and policies.

Acknowledgements This work was partially funded by the BMWi project CloudCycle (01MD11023). We thank Kálmán Képes for his work on the prototype.

References

1. Binz, T., et al.: OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In: ICSOC. Springer (2013)
2. Binz, T., et al.: TOSCA: Portable Automated Deployment and Management of Cloud Applications, pp. 527–549. Advanced Web Services, Springer (Januar 2014)
3. Kopp, O., et al.: Winery – Modeling Tool for TOSCA-based Cloud Applications. In: ICSOC. Springer (2013)
4. OASIS: OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0 Committee Specification 01 (2013)
5. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0 (January 2013)

[3] `http://www.moodle.org`

What About Database-centric Enterprise Application Integration?

Daniel Ritter

HANA Platform, SAP AG
Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
daniel.ritter@sap.com

Abstract. The focus on "big data" and the emergence of hybrid Online Transaction Processing/Online Analytical Processing systems in the database community creates new opportunities for (business) application vendors. More and more business logic is "pushed" to the database, i. e., close to the application data, for faster and more efficient processing, while avoiding unnecessary data shipments.
In this position paper, we argue that *Enterprise Application Integration* should engage in a liaison with the recent data-processing advances, especially for supporting the integration of applications running in the database with remote applications.

Keywords: Enterprise Application Integration, Relational Database.

1 Introduction and Position

The recent advances within the database research community–not limited to hybrid transactional and analytical systems (e. g., [5,9])–bring applications together on one platform: the database. For less data shipment and more efficient processing, (business) application logic is "pushed" to the databases by translating it to standard SQL, PL/SQL and an increasing number of additional libraries and programming languages [1,2]. The databases of complete business application suites[1] are not only used for "bookkeeping" anymore, but transform to application platforms. The conventional application systems remain the presentation layer, while the control and data flows move closer to the databases.

These applications require message-based integration. Some of the data and message endpoints like *Data Stream and Complex Event Processing* (CEP), for *Information Flow Processing* [4], and *ETL* (e. g., *Microsoft SQL Server Integration Services*) already operate on database level. While these data endpoints care about high-performance inbound data loading, (message) protocol and format-level transformations (e. g., JSON, XML to relational model), data cleansing and storage to database tables, standard integration capabilities like routing, mapping, and guaranteed delivery are left for EAI systems, which are still implemented on application server level. When additionally considering the subsumption premises

[1] For example, the SAP Business Suite on HANA: https://www.suiteonhana.com

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

of “Too much Middleware" [11] (i. e., discussing the costs/benefits of a separate EAI system) and the requirements to middleware systems that are already well-covered by most database systems (e. g., scalability, data consistency/transaction processing, user management, high availability), we target the question whether message-based integration is viable from a database perspective. The required system shall realize EAI solely using standard relational processing (e. g., SQL, PL/SQL) for an efficient evaluation of integration semantics. We address the following research questions, discussed subsequently: (1) “Can integration logic (e. g., mapping, routing, aggregation [7]) be “pushed" into the database completely?", (2) “For which *Integration Scenarios* does that bring which benefits?", and (3) “What are its advantages and disadvantages?"

2 The Database as Integration Middleware

We found recent evidence to our position in the areas of declarative message processing in XQuery/XML-DBMS (e. g., [10,3]) and *Publish/Subscribe* in relational databases [6]. Since this work considers only a small subset of integration semantics, requires massive extensions to standard SQL and most business application data is stored in relational databases, these approaches only support its overall position. Subsequently, we briefly sketch the most relevant aspects, namely *Integration Cases* and *Common EAI Scenarios*.

Table 1. Relevant *Integration Cases* considered for database and integration systems (Middlew.) based on control and data flow. Relevant case for this paper is **highlighted**.

Cases	Data Flow	Control Flow	Comment
Case 1	Middlew.	Middlew.	no persistent storage of messages; synchronous messaging
Case 2	**Database**	**Database**	**Database with Protocol Adapters (no Middlew.)**
Case 3	Database	Middlew.	Middlew. controls execution on Database and its Adapters
Case 4	Middlew./Database	Middlew./Database	Shared control and data flow
Case 5	Middlew./Database	Middlew.	Middlew. controls shared data processing
Case 6	Middlew./Database	Database	Database controls shared data flow

Integration Cases For a systematic definition of the term “database-centric EAI" we distinguished six different and relevant integration cases between the poles of control and data flow for database and integration systems. These cases are listed in Table 1. We focus on *Case 2*, in which the database handles the control and data flow execution exclusively, while no additional application tier for the middleware system (short “Middlew.") is required. In general, EAI systems consist of protocol adapters (i. e., message endpoints) and integration logic. The endpoints are assumed to be available on database-level (e. g., CEP, ETL on inbound and messaging [6] on inbound and outbound site). The integration logic is expressed as standard database artifacts like SQL, PL/SQL. For comparison, we defined *Case 1* as integration system without storage (e. g., no asynchronous

messaging possible) and *Case 5* refers to current EAI systems with persistent message storage (e. g., Java Messaging Service (JMS)[2] based processing) and control over message processing and data flows.

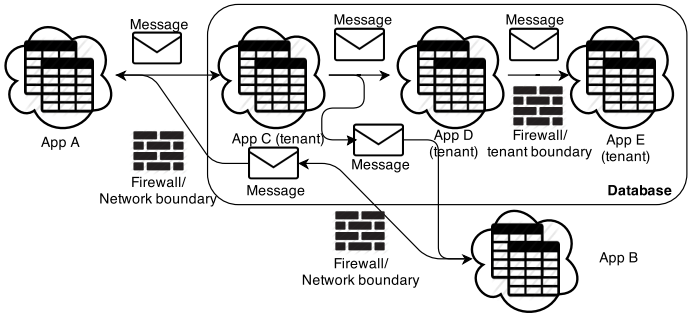


Fig. 1. Common EAI Scenarios: C1 (*App C* to *App D*), C2 (in: *App A* to *App C*, out: *App C* to *App B*) and C3 (*App A* to *App B* via *database*). In case of cross-tenant communication (*App D* to *App E*) we assume equivalence to C2 (outbound/inbound).

Common EAI Scenarios Figure 1 shows an overview of common EAI scenarios that are combined to categories *C1-3*. Category *C1* (local/local: from *App C* to *App D*) refers to the integration of a sender and a receiver application residing in the same application system, sharing the same database, but not the schema (e. g., as in business suites). Category *C2* (local/external: inbound processing from *App A* to *App C*; outbound processing from *App C* to *App B*) covers the cases, in which an application on the database (sender or receiver) needs to access or receive external data (e. g., access remote applications). Finally Category *C3* (external/external: from (*App A* to *App B*) features sender and receiver applications residing in different application systems, while communication is mediated by a dedicated "integration database system" (i. e., corresponds to the classical middleware case). The "multi-tenant database" case from [8] is depicted by the communication from *App D* to *App E*, which requires an even stricter tenant and schema separation that eventually translates to C2. As example for a C2 scenario, we selected the SAP ERP *Convergent Invoicing* (FI-CA) use case. For the invoicing process, high-volume billable items (up to 500k per second– approx. 4 billion messages per day) wrapped as messages (e. g., one iTunes song or telephone call decomposes to records for customer, vendor, author etc) are created and sent by the *Convergent Charging* application. The messages have to be validated, enriched with master data, filtered and aggregated according to multi-dimensional and potentially customer specific criteria, before legally binding documents are created. JMS messaging could queue that amount of messages[3], however, cannot handle the integration logic.

[2] Java Messaging Service: http://www.jcp.org/en/jsr/detail?id=914
[3] JMS benchmark: http://www.spec.org/jms2007/

3 Discussion

The research questions formulated in Section 1 (i. e., (1)–(3)) target the feasibility and viability of a database-only integration approach (see Case 2, Section 2). The question about integration logic “push-down" to the database system (1) was approached experimentally by a prototypical implementation. The evaluation showed short-comings in terms of language expressiveness (e. g., timed-aggregations are not possible), the need for a scheduled pipeline processing, when a (transactional) decoupling between sender and receiver is required, and a small latency penalty when mixing SQL and PL/SQL processing. Besides that, the processing of bulks of messages showed results for throughput comparable to the requirements, e. g., of the FI-CA scenario, thus leading to the question about viable scenarios (2). The natural category for database-centric processing seems to be C1 for database local, schema-to-schema integration. However, this is currently completely covered by “shared-schema" integration, although it could untie the tight application coupling. The support of database adapters with pre-/post-protocol conversion integration logic processing C2 seems most promising (see FI-CA scenario). Category C3 is the domain of current EAI systems and only seems to show benefits for scenarios with qualities that require frequent persistent message storage and/or massive data-bound computations for database-only integration. The benefits and additional operational qualities (3) are stable asynchronous and transactional message processing, portable database code (for standard SQL logic only), interoperability through database protocol adapters, and less frequent and expensive data format conversions.

References

1. C. Binnig, N. May, and T. Mindnich. SQLScript: Efficiently Analyzing Big Enterprise Data in SAP HANA. In *BTW*, pages 363–382, 2013.
2. C. Binnig, R. Rehrmann, F. Faerber, and R. Riewe. Funsql: it is time to make sql functional. In *EDBT/ICDT Workshops*, pages 41–46, 2012.
3. A. Böhm, C.-C. Kanne, and G. Moerkotte. Demaq: A foundation for declarative xml message processing. In *CIDR*, pages 33–43, 2007.
4. G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15, 2012.
5. F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. The SAP HANA Database – An Architecture Overview. *IEEE Data Eng. Bull.*, 35(1):28–33, 2012.
6. D. Gawlick and S. Mishra. Information sharing with the oracle database. In *DEBS*, 2003.
7. G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
8. D. Jacobs and S. Aulbach. Ruminations on multi-tenant databases. In *BTW*, pages 514–521, 2007.
9. A. Kemper and T. Neumann. One size fits all, again! the architecture of the hybrid oltp&olap database management system hyper. In *BIRTE*, pages 7–23, 2010.
10. N. Onose and J. Siméon. Xquery at your web service. In *WWW*, pages 603–611, 2004.
11. M. Stonebraker. Too much middleware. *SIGMOD Record*, 31(1):97–106, 2002.

Towards Process-based Composition of Activities for Collecting Data in Supply Chains

Gregor Grambow, Nicolas Mundbrod, Vivian Steller, and Manfred Reichert

Institute of Databases and Information Systems
Ulm University, Germany
{gregor.grambow,nicolas.mundbrod,vivian.steller,manfred.reichert}@uni-ulm.de
http://www.uni-ulm.de/dbis

Abstract. Manufacturing companies more and more face the challenge of ensuring sustainable production. In particular, they continuously need to report sustainability data about their products and manufacturing processes that is categorized by various sustainability indicators. However, in a supply chain, such data collection also involves the companies suppliers. Thus, companies must issue cross-organizational data collection processes with potentially high numbers of responders. Due to the heterogeneity in a supply chain and the necessary involvement of services from external sustainability service providers, such processes are often long-running and error-prone. In response to that, we propose an approach for automatically and contextually assembling the required activities and services and managing them by an explicitly specified and enacted process.

Keywords: Process Configuration, Business Process Variability, Data Collection, Sustainability, Supply Chain

1 Introduction

Nowadays, companies collaborate in supply chains in order to assemble complex products like cars or electronic devices. Such companies face a specific challenge: state authorities and the market require sustainable production. Therefore, companies are increasingly forced to report sustainability data about their production that is classified by sustainability indicators like, for example, greenhouse gas (GHG) emissions or the amount of lead contained in products. However, to report respective data, in turn, a company must request it from its suppliers. In general, a sustainability data request could be passed through multiple levels of the supply chain as illustrated in Figure 1.

Sustainability data requests involve great heterogeneity: different companies follow different approaches in sustainability data management. Some of them have different in-house systems (IHSs) for this (cf. S1.1 in Figure 1), whereas others rely on manual management (cf. S1.3) or even have no proper approach to it at all. Furthermore, in various cases, services of external service providers may be required. For example, a company reporting GHG emissions in production (cf.

N. Herzberg, M. Kunze (eds.): Proceedings of the 6th Central European Workshop on Services and their Composition (ZEUS 2014), Potsdam, Germany, 27-03-2014, published at http://ceur-ws.org

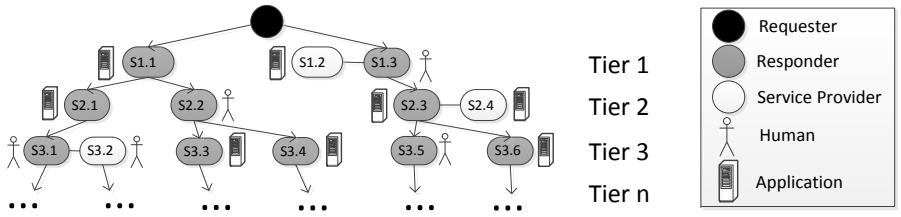


Fig. 1: Supply Chain Data Collection

S1.3) might need an external service provider to validate the data (cf. S1.2) after having collected their own data and received relating data from its suppliers (cf. S2.3).

Due to these properties, data collection process can be long-running, tedious, and error-prone, which may even involve legal fines for the reporting companies. Due to the heterogeneity in tools and approaches, it is not possible to apply a federated information system or data base to all companies. Furthermore, as such data exchange not only involves the mere exchange of data but involves various kinds of activities relating to such data, simple data integration approaches, as e.g., utilizing ontologies are also not sufficient. The following scenario gives a small-scale industrial example for such a data collection process.

Scenario: Sustainability Data Collection
An automotive company wants to collect sustainability data relating to a specific part. In particular, a regulation requests the reporting of the quantity of lead in that part. This also concerns sub-parts of that part that are delivered by two suppliers of the company. One of them is a bigger company with a IHS in place. The other one is a smaller company with no system and no dedicated responsible for sustainability. The IHS of the bigger company has its own data format that has to be explicitly converted to be useable. For the smaller company, a service provider is needed that will validate the manually collected data to ensure that it complies with legal regulations. This simple scenario already shows how much complexity can be involved even in simple requests and gives an outlook on how this can look like in bigger scenarios involving hundreds or thousands of companies with different systems and properties.

In the SustainHub project[1], we are developing a centralized information exchange platform (also called SustainHub) that supports sustainability data collection along the entire supply chain. For this purpose, we have investigated and discussed the challenges for sustainable supply chain communication as well as the state-of-the-art [4].

As this paper focuses on the composition of activities and services, first of all we summarize the challenges a system must tackle to enable this. (DCC1) Most of

[1] SustainHub (Project No. 283130) is a collaborative project within the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

the activities in sustainability data exchange are still executed manually. Taking into account that such data exchange takes place in complex supply chains, it can be problematic to even find the right person in the right department in the right company or the right service of the right service provider. To enable automated support, such information must be explicitly stored and managed. (DCC2) Different companies have different ways to manage relevant sustainability data. Some use IHSs, whereas others rely on manual data management. A system supporting data collection in a supply chain, therefore, must be able to access it in both ways, i.e. it must support manual as well as automated data collection. (DCC3) The requests in sustainability data exchange rely on a myriad of different factors (e.g., legal requirements, IHS used in a company). To support repeatable data collection, a system must be aware of such contextual factors and manage them centrally. (DCC4) Due to the great number of different factors, each request is not far from being unique and has to be executed manually. A system supporting such data exchange should enable the centralized definition of the data collection process and also its different variants. Both definitions should be as simple as possible to not burden users with a cumbersome and error-prone modeling process.

The remainder of this paper is organized as follows: Section 2 presents our approach for sustainability data collection and exchange. Section 3 gives a brief overview about related work followed by a conclusion.

2 Automated Process for Data Collection

Basically, our approach for supporting the complex process of sustainability data collection involves the idea to govern that process by an explicitly specified process, which is automatically enacted by a Process-Aware Information System (PAIS). That way, each request can be managed in a centralized way (cf. DCC1) and be specified explicitly (cf. DCC4) via a process template. Furthermore, this approach bears another advantage: For the different activities in such a process, custom components can be applied. These components can be used for manual activities as well as connections to different IHSs. In addition, other components can realize services of various service providers. This facilitates support for manual and automatic data collection (cf. DCC2). Further, makes the processes modular and reusable.

However, such an approach would involve rigidly pre-specified process templates and still no awareness of meta data like contextual factors. A large number of process templates would have to be specified in advance incorporating every possible combination of eventualities. A human would then have to select the right one being aware of each and every parameter of the current request. This would be tedious and error-prone. In response to this, we have created an approach capable of automatically incorporating contextual factors into a system and, based on them, creating various variants of pre-specified processes exactly matching the current request situation. Figure 2 illustrates the different components of this approach.

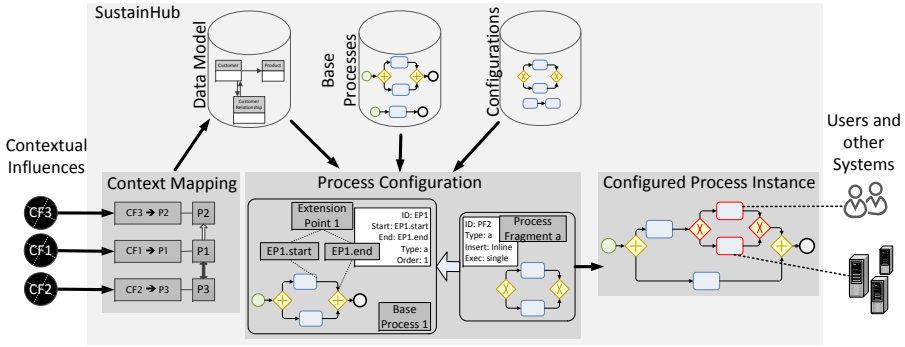


Fig. 2: Automated Process and Service Configuration

To be able to automatically process contextual factors and to utilize them for the automatic creation of process variants, our approach applies a set of different components: to incorporate contextual factors, the 'Context Mapping' component maps these to parameters directly usable for process configuration. The latter is applied by the 'Process Configuration' component that creates process variants (cf. DCC4) based on two entities: base processes incorporating the basic activities necessary for a specific type of request and process fragments depicting sets of activities suitable for a specific situation. By combining of these two, a configured process instance is created, as shown in Figure 2.

However, to be able to automatically apply such configurations, the 'Process Configuration' component must be aware of various facts and their relations. This includes data about the companies connected to SustainHub to be able to deliver activities to the right persons or IHSs as well as basic domain data accessible for all companies, like sustainability indicators. Furthermore, the actual data exchange (e.g. the data of a request) and the content exchanged must be stored and available to SustainHub as well. All this data must be mutually connected as well as connected to other data serving as basis for automatically managing context data and process configuration and enactment. Therefore, we apply a data model that is more comprehensive as those usually applied in PAIS integrating types of data usually found in other systems (e.g., ERP systems). We will not explain all entities here, however, we will introduce six sections of it containing entities for different purposes: 'customer data' (e.g. organizational models or IHS references), 'master data' (e.g. indicators or substances), 'runtime data' (e.g. request data), 'content data' (e.g. values actually exchanged), 'context data' (contextual factors), and 'process data' (e.g. data necessary for managing and configuring processes automatically). Having all the data in place, we now go into detail about the two main components for context mapping and process configuration.

2.1 Context Mapping

As already stated, variants of sustainability data requests can depend on a myriad of contextual factors. Consequently, a system enabling automated management of

such variants must apply a consistent way of managing and storing such factors (cf. DCC3). Moreover, there is often no one-to-one mapping between a factor and a variant or a certain set of activities. For example, a company might apply a special four-eyes-principle approval process in two cases. The first involves data relating to a specific law and involving high fines. The second concerns data relating to a specific customer group that the company has no high trust in. Enabling variant management by creating one rule to apply one certain variant (or adaptation) to a base process would result in a high number of rules. This would bloat the needed data and make modeling and maintaining cumbersome. To avoid this, we apply a simple and lightweight mapping of contextual factors to process parameters that can be directly used for the process variant configurations. As illustrated in Figure 2, our approach features simple logical mapping rules (e.g. $CF1 \wedge CF2 \rightarrow P1$) and also the option to apply simple consistency rules to avoid erroneous configurations (e.g. P1 and P3 mutually exclude each other). A simple example for a context factor as shown in the scenario in the introduction would be the approach and tool a company uses for sustainability data management. This can be mapped to concrete process parameters, e.g., that the company uses a specific IHS for a specific data collection task. That way, a distinct set of parameters for the selection of configurations can be created.

2.2 Process Configuration

When a stable set of parameters is in place for a specific request, it must be determined what exactly shall be inserted into its base process and where to insert it. This requires that options for both of these decisions are available in the variant model of SustainHub. As stated in DCC2, both the definition of the base processes and the configurations should be as simple as possible to not burden the users with a complicated modeling process. Therefore, we aimed for a simple and lightweight way of modeling.

Our studies have shown that for most requests, a basic set of activities is mandatory, e.g. configuration of the data collection or the final data delivery. Therefore, we have decided to allow for the modeling of base processes with mandatory activities for different cases (as e.g. sustainability indicators) and to only extend these processes with additional activities instead of also applying deletions. These base processes are then annotated with *extension points* to indicate where an extension is feasible. As illustrated in Figure 2 such extension points have two connections with the process to clearly indicate, where the insertion should happen and a set of meta information, SustainHub can then use to determine, which fragment would be suitable at that position. Furthermore, a set of options is used to determine, if fragments should be inserted as sub-process or directly in the base process (inline) and in which order they are to be inserted if multiple extension points are at the same position.

Our studies showed that a specific set of activities is often cohesively needed in a specific situation (e.g. if data has to be collected manually, that activity involves also other activities like informing the responsible person). Therefore, we have decided to apply whole fragments instead of fine-grained adaptations adding

single activities. Moreover, the approach becomes easier to model and maintain that way as we allow to model such fragments same as the base processes in a PAIS. That way, that PAIS manages their structural correctness and other basic factors.

2.3 Example Scenario

In this section, we show the application of our approach to the concrete industrial example scenario applied in the introduction. Figure 3 illustrates this including a base process, different extension points, context factors, process fragments, and a resulting configured process. The base process comprises three activities for configuring the request, aggregating the results, and delivering them to the requester (cf. Figure 3). It also has two extension points, EP1 for the data collection activities, EP2 for post processing activities. Via a specific parameter (Order) it is ensured, that EP1 fragments are inserted before EP2 fragments. Furthermore, various fragments are in place. In Figure 3, four of them are shown, for automatic and manual data collection as well as for external validation of manually collected data and for conversion of automatically collected data.

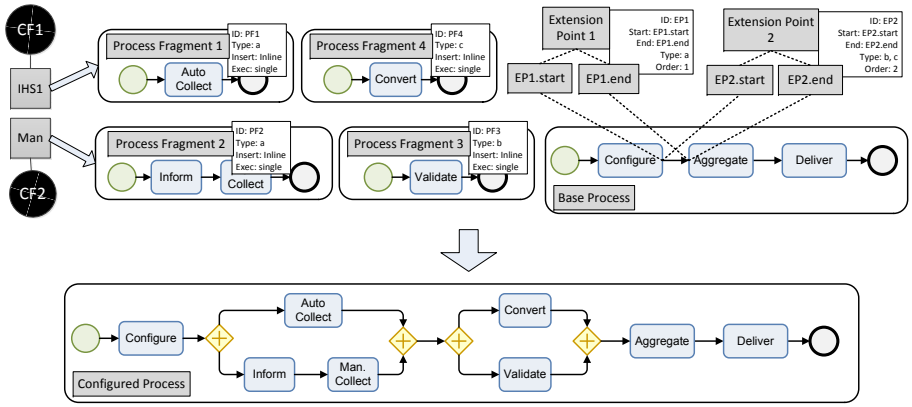


Fig. 3: Process Configuration Scenario

For this request, the system has the facts in place, that two responders are involved and that one of them has an IHS while the other relies on manual data collection (for which the responsible person is also modeled within SustainHubs' data model). Thus, these context factors can be mapped to two process parameters indicating one responder with different properties each. Each of the parameters is configured to imply two fragments, one for data collection, one for post processing. These fragments are then automatically integrated into the base process, all of them inline, as configured. Fragment 1 and 2 are integrated in parallel via EP1 and fragment 3 and 4 also via EP2. Finally, the resulting configured process is shown in the lower section of Figure 3.

3 Related Work

Our approach presented in this paper enables the automated assembly of various activities and services necessary for sustainability data requests applying contextually configured processes for that. Therefore, and due to the lack of space, we limit our review of related work to process configuration approaches. Examples include ADOM [6] that relies on software engineering principles or configuration modeling approaches like C-EPC [7] or C-YAWL [3]. Like the most other approaches, they focus solely on the modeling of process configuration. They take different approaches to configuration: ADOM enables the specification of guidelines and constraints, while C-EPC integrates configurable elements into the model. C-YAWL allows for hiding single elements or even blocking whole execution paths. For a qualitative comparison of such configuration approaches, see Ayora et al. [1]. Besides the fact that such approaches only focus on modeling, they also require human interaction for manual application of the configuration in contrast to our approach.

In addition to this, other approaches like [8] target the correctness of process configurations. In contrast to them, our approach encapsulates the minimal set of necessary activities into a base process and other sets of activities in process fragments. Both of these are modeled in a PAIS and can be checked for correctness therein. Furthermore, we limit the complexity of the configurations with the explicit extension points. That way, our approach becomes more lightweight and easy to handle.

Provop [5] constitutes an approach that is more closely-related to our approach. It enables the modeling of base processes and pre-specified configuration options and also the execution of processes configured that way. However, it is more fine-grained and complicated as our approach and it does not allow completely automatic context acquisition, processing, and process configuration. For a broader view on related work, see our paper on challenges and state-of-the-art [4].

4 Conclusion

In this paper, we have introduced an approach for applying configurable processes for complex data collection scenarios like sustainability data exchange in supply chains. The approach is lightweight featuring pre-specified base processes and allows for configuring these with also pre-specified process fragments to adhere to the specifics of various different situations. Moreover, our approach enables such configurations to be applied automatically involving techniques for acquiring, storing, and managing various contextual factors to which the processes must comply.

Our future work will involve the extension of our approach to satisfy all requirements we have specified in [4]. This involves runtime adaptations to processes (e.g., for situations, in which the context changes while a request is already processed) as well as monitoring and quality management capabilities. Furthermore, we have already started to implement our approach on top of the

AristaFlow BPM suite [2] and have also begun to evaluate it using 66 sustainability indicators we have collected from industry surveys in the SustainHub project.

Acknowledgement

The project SustainHub (Project No. 283130) is sponsored by the EU in the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

References

1. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Enhancing modeling and change support for process families through change patterns. In: 14th Int'l Working Conference on Business Process Modeling, Development, and Support (BPMDS'13). pp. 246–260. No. 147 in LNBIP, Springer (June 2013), http://dbis.eprints.uni-ulm.de/930/
2. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development 23(2), 81–97 (2009), http://dbis.eprints.uni-ulm.de/486/
3. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable workflow models. Int. J. Cooperative Inf. Syst. 17(2), 177–221 (2008)
4. Grambow, G., Mundbrod, N., Steller, V., Reichert, M.: Challenges of applying adaptive processes to enable variability in sustainability data collection. In: 3rd Int'l Symposium on Data-Driven Process Discovery and Analysis. pp. 74–88 (2013)
5. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: The provop approach. Journal of Software Maintenance and Evolution: Research and Practice 22(6-7), 519–546 (November 2010), http://dbis.eprints.uni-ulm.de/628/
6. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Extending the adaptability of reference models. IEEE Trans on Syst, Man, and Cyber, Part A 40(5), 1045–1056 (2010)
7. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Information Systems 32(1), 1–23 (2005)
8. Van Der Aalst, W., Lohmann, N., La Rosa, M., Xu, J.: Correctness ensuring process configuration: An approach based on partner synthesis. In: Business Process Management, pp. 95–111. Springer (2010)

Author Index