ZEUS 2017

9th Central European Workshop on Services and their Composition

Oliver Kopp, Jörg Lenhard, Cesare Pautasso

USI Lugano, Switzerland, 13-14 February 2017

Onsite Proceedings

Oliver Kopp
Jörg Lenhard
Cesare Pautasso

ZEUS 2017

9th ZEUS Workshop, ZEUS 2017,
Lugano, Switzerland, 13–14 February 2017
Onsite Proceedings

Volume Editors

Oliver Kopp
University of Stuttgart, Institute for Parallel and Distributed Systems
Universitätsstraße 38, DE-70569 Stuttgart
oliver.kopp@ipvs.uni-stuttgart.de

Jörg Lenhard
Karlstad University, Department of Mathematics and Computer Science
Universitetsgatan 2, SE-65188 Karlstad
joerg.lenhard@kau.se

Cesare Pautasso
USI Faculty of Informatics,
Architecture, Design and Web Information Systems Engineering Research Group
Via Buffi 13, CH-6900 Lugano
c.pautasso@ieee.org

Preface

We welcome all workshop participants to the 9th edition of the ZEUS Workshop in Lugano. This workshop series offers young researchers an opportunity to present and discuss early ideas and work in progress as well as to establish contacts among young researchers. For this year's edition, we selected 10 regular submissions, four position papers, and one tool demonstration by researchers from Austria, Brasil, Germany, Italy, Switzerland and the United Kingdom for presentation at the workshop. Each submission went through a thorough peer-review process and was assessed by at least three members of the program committee with regard to its relevance and scientific quality. The contributions that were accepted for presentation at the workshop covered the areas of Process Analysis, Process Enactment and Modeling Languages, Stream Processing, the Internet of Things, Cloud Management, and Software Modeling and Analysis. In addition, the workshop will host a mini-tutorial on support for literature studies with the JabRef reference manager.

The workshop program is further enriched by two keynotes. The first keynote is held by Daniel Lübke on the topic *Why developers don't like BPM and how research can help* discusses the perception of the BPM field by developers in industry. The second keynote titled *From Service- to UI-Oriented Computing: The Vision of an Intuitive Composition Paradigm* is given by Florian Daniel, who presents a composition paradigm that leverages the UIs of applications, instead of their APIs, and that makes composition-based development accessible to an ever wider range of developers.

The workshop is generously sponsored by innoQ Deutschland GmbH.

Lugano, February 2017

Oliver Kopp
Jörg Lenhard
Cesare Pautasso

Organization

Steering Committee

Oliver Kopp	University of Stuttgart
Jörg Lenhard	Karlstad University
Christoph Hochreiner	TU Wien

Local Organizer

Cesare Pautasso	USI Lugano

Program Committee Chairs

Oliver Kopp	University of Stuttgart
Jörg Lenhard	Karlstad University
Cesare Pautasso	USI Lugano

Program Committee

Saimir Bala	Vienna University of Economics and Business
Anne Baumgrass	Synfioo – 360° Transportation Monitoring
Domenico Bianculli	University of Luxembourg
Daniele Bonetta	Oracle Labs
Richard Braun	TU Dresden
Dirk Fahland	Eindhoven University of Technology
Alessio Gambi	Saarland University
Matthias Geiger	University of Bamberg
Georg Grossmann	University of South Australia
Thomas Heinze	University of Jena
Nico Herzberg	SAP SE
Christoph Hochreiner	TU Wien
Conrad Indiono	University of Vienna
Meiko Jensen	ULD Schleswig-Holstein
Stefan Kolb	University of Bamberg
Oliver Kopp	University of Stuttgart
Agnes Koschmider	Karlsruhe Institute of Technology
Philipp Leitner	University of Zurich
Jörg Lenhard	Karlstad University
Henrik Leopold	VU University Amsterdam
Stephan Reiff-Marganiec	University of Leicester
Andreas Schoenberger	Siemens AG
Stefan Schulte	TU Wien
Jan Sürmerli	Humboldt University of Berlin
Matthias Weidlich	Humboldt University of Berlin
Matthias Wieland	University of Stuttgart

Subreviewers

Johannes Wettinger	University of Stuttgart
Peter Reimann	University of Stuttgart
Han van der Aa	VU University Amsterdam
Michael Zimmermann	University of Stuttgart
Felix Baumann	University of Stuttgart

Sponsoring Institutions

innoQ Deutschland GmbH

Contents

Towards Certified Data Flow Analysis of Business Processes

Thomas S. Heinze

Friedrich Schiller University Jena, 07743 Jena, Germany
t.heinze@uni-jena.de

Abstract. Data flow analysis allows for the static analysis of business processes. Certified data flow analysis would even allow for a trustwhorty analysis, as the analysis comes with a machine-checkable correctness proof. In this paper, we argue for a certified analysis of business processes.

1 Introduction and Motivation

Data flow analysis has shown to be a utility for business process modeling and analysis, be it for supporting formal verification [5,6], for analyzing data- or control-flow related process properties [2,11], or for optimization and re-engineering [8]. In particular two aspects make it such a useful method: (1) Data flow analysis provides a general framework which can be easily adjusted to new domains and analysis problems, (2) data flow analysis is grounded on well-founded theories like Kildall's lattice-theoretic formulation [7] or Cousot's *abstract interpretation* [4]. While the latter aspect offers a way to formally show an analysis to be correct, until now, only a small number of data flow analyses for business processes have been proven correct, mostly using pencil-and-paper proofs and just considering termination [2,5]. This is suprising considering the recent advances made in the verification of static analysis. State-of-the-art approaches define not only an analysis but rather add a mechanized, that is machine-checkable proof of its correctness. The term *certified analysis* has therefore been coined, as a user does not need to trust in such an analysis but can automatically check its conjoined correctness proof. Most notably, the verification of an optimizing C compiler [9] documented the power of the certified analysis approach.

In this position paper, we argue for the idea of a *certified analysis of business processes*. We believe the advantages to be manifold. For instance, the complexity of business process modeling languages like BPEL and BPMN make not only the design and implementation of processes, but also of analyses error-prone. An approach for proving analysis correctness thus helps in regaining confidence. This can be of vital importance considering, e.g., business processes in healthcare. Further, a mechanized correctness proof of a compliance analysis augments process auditing scenarios [1] with an orthogonal check that the audit itself can be trusted. Synthesizing an analysis from its correctness proof, even relieves the analysis designer from the implementation burden. In the following, we will introduce the concept of certified analysis based upon abstract interpretation, sketch challenges which arise when applied to business processes and finally outline our first steps towards a certified data flow analysis of business processes.

2 Abstract Interpretation and Certified Analysis

Data flow analysis can be seen as *abstract interpretation* [4], where programs are evaluated using abstract values instead of concrete values. As an example, instead of performing arithmetic operations on integers when executing a program, an analysis may only track whether an integer has abstract value *odd* or *even*. In this way, e.g., addition boils down to four cases: $\mathit{even}+\mathit{even}=\mathit{even}$, $\mathit{even}+\mathit{odd}=\mathit{odd}$, $\mathit{odd}+\mathit{even}=\mathit{odd}$, and $\mathit{odd}+\mathit{odd}=\mathit{even}$. The domains of abstract and concrete values are connected by the concretization function γ, which maps abstract values to the represented concrete ones, e.g., $\gamma(\mathit{odd}) = \{1, 3, 5, \dots\}$. In addition, the abstract domain A should form a partially-ordered set, reflecting precision among abstract values such that $\forall a, b \in A\colon a \leq b \leftrightarrow \gamma(a) \subseteq \gamma(b)$, and contain a distinct *top* element with $\forall a \in A\colon a \leq \mathit{top}$, representing all concrete values. Executing a program's statements on concrete values can be formalized using Hoare's weakest precondition calculus, as in [3], which yields the concrete semantics. The effect of statements on abstract values is defined by the analysis in terms of a monotone function $f\colon A \to A$ mimicing, e.g., above's addition example, which in this way provides the abstract semantics. An analysis based upon abstract interpretation then calculates a fixpoint abstract value, i.e., $f(a) = a$, for a given program by continuously applying the abstract semantics to an initial abstract value until the fixpoint is reached (optionally using accelerators like widening/narrowing [4]).

An analysis can then be shown correct based upon abstract interpretation, iff:

$$\forall a \in A\colon c_i \in \gamma(a) \to f(a) = a \to \forall c'\colon c_{\mathit{initial}} \rightsquigarrow c' \to c' \in \gamma(a)$$

meaning that a fixpoint a of the abstract semantics, that includes the initial concrete value c_i of the program in its concretization, also includes the concrete values c' of all reachable program states, $c_{\mathit{initial}} \rightsquigarrow c'$, in its concretization. Note that correctness here refers to soundness, i.e., the abstract value a is guaranteed to (over-)approximate the concrete value. The *Coq proof assistent*[1] has been shown of use for proving analysis correctness [3,9]. Its inductive types provide a way for encoding programs and its recursive functions for encoding concrete and abstract semantics. Proving analysis correctness in Coq may still require manual work, yet Coq allows for automatic proof validation and even to synthesize the analysis implementation[1], which justifies the notion of a *certified analysis*.

3 Research Challenges and First Steps

While the automated analysis for certifying business processes has been a recent research topic [1], we are not aware of any prior work on the certification of such an analysis for business processes itself. As indicated above, though, the area of static program analysis provides a rich spectrum of methods for proving analysis correctness. Yet, business processes and business process modeling languages feature certain peculiarities which challenge a certified analysis approach:

[1] `https://coq.inria.fr/` (link was last followed on February 1, 2017)

– First, as parallelism is essential to business processes, the concrete and abstract semantics need to reflect parallel executions. Application of *(concurrent) separation logic* [10] can thus be of advantage, extending the Hoare calculus with a rule for independently reasoning about parallel processes.
– Second, the phletora of language constructs in languages like BPEL or BPMN make formalizing the concrete semantics a tedious task. We thus opt for an approach, where we consider a *kernel* of basic language constructs, mapping all other constructs onto the kernel. This is a common method to boil down the complexity of a language [2] and furthermore provides a starting point for coping with the different and often mixed types of data flow definitions (XPath, Java, scripting languages, etc.) used in BPMN processes.
– Third, unstructured processes. While unstructuredness in case of sequential control flow alone can be handled by basing semantics on low-level jump operations instead of high-level ifelse or loop statements, mixed use of gateways as possible in BPMN, even more in the presence of the notoriously difficult to formalize (inclusive) OR gateway, provides for a real research challenge.

Currently, we are formalizing a toy language for business processes in Coq, supporting basic workflow patterns, e.g, sequence, parallel split/synchronize, exclusive choice/merge, and a data flow analysis for computing communication dependencies (see [2]). This language shall be enriched in future steps. Future work will also address other analyses, where we want to follow a modular approach using a general analysis framework, which is instantiated for a specific analysis.

References

1. Accorsi, R., Lowis, L., Sato, Y.: Automated Certification for Compliant Cloud-Based Business Processes. BISE 3(3), 145–154 (2011)
2. Amme, W., Martens, A., Moser, S.: Advanced verification of distributed WS-BPEL business processes. IJBPIM 4(1), 47–59 (2009)
3. Bertot, Y.: Structural Abstract Interpretation. In: Language Engineering and Rigorous Software Development, LNCS, vol. 5520, pp. 153–194. Springer (2008)
4. Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs. In: POPL'77, Proc. pp. 238–252. ACM (1977)
5. Heinze, T.S., Amme, W.: Sparse Analysis of Variable Path Predicates Based upon SSA-Form. In: ISoLA'16, Proc. (1). LNCS, vol. 9952, pp. 227–242. Springer (2016)
6. Heinze, T.S., Amme, W., Moser, S.: Compiling More Precise Petri Net Models for an Improved Verification of Service Implementations. In: SOCA 2014, Proc. pp. 25–32. IEEE (2014)
7. Kildall, G.A.: A Unified Approach to Global Program Optimization. In: POPL'73, Proc. pp. 194–206. ACM (1973)
8. Kopp, O., Khalaf, R., Leymann, F.: Deriving Explicit Data Links in WS-BPEL Processes. In: SCC 2008, Proc. (2). pp. 367–376. IEEE (2008)
9. Leroy, X.: Formal Verification of a Realistic Compiler. Comm. of the ACM 52(7), 107–115 (2009)
10. O'Hearn, P.W.: Resources, Concurrency and Local Reasoning. In: CONCUR 2004, Proc. LNCS, vol. 3170, pp. 49–67. Springer (2004)
11. Saad, C., Bauer, B.: Data-Flow Based Model Analysis and Its Applications. In: MODELS 2013, Proc. LNCS, vol. 8107, pp. 707–723. Springer (2013)

Automatic Standard Compliance Assessment of BPMN 2.0 Process Models

Matthias Geiger[1], Philipp Neugebauer[2], and Andreas Vorndran[1]

[1] Distributed Systems Group, University of Bamberg, Germany
matthias.geiger@uni-bamberg.de
[2] innoQ Deutschland GmbH, Kreuzstraße 16, 80331 München, Germany

Abstract. The *Business Process Model and Notation 2.0* is nowadays the de-facto standard for process and workflow modeling. It is supported by many modeling tools and engines which are able to consume and execute processes modeled in the native BPMN 2.0 syntax. Despite its popularity, there are still issues and drawbacks regarding standard compliance of BPMN 2.0 process models. This paper elaborates on reasons for such compliance problems and describes how those compliance issues can be revealed by performing automated checks. Finally, it is shown that standard compliance is still an issue by analyzing a set of process models.

Keywords: BPMN 2.0, standard compliance, static analysis

1 Introduction

Since its publication by the Object Management Group (OMG) in 2011, the *Business Process Model and Notation* (BPMN) 2.0 [15] is the de-facto standard for process and workflow modeling, both accepted in academia and practice. This is also reflected by its adoption as an ISO/IEC standard in 2013 [10]. BPMN 2.0 is used in a wide range of scenarios ranging from simple visualization purposes on a classic whiteboard, over documentation and simulation to actual execution of modeled processes using dedicated BPMN engines. Depending on the scenario, there is an increasing need for *valid*, standard compliant process models and model serializations, as these are a prerequisite for sensible results, especially for process simulation and execution [6, 13].

The specification [10] defines which visual shapes can be used for modeling, which constraints have to be respected and finally, since 2.0, how a model should be serialized in a standardized XML-based format. However, due to the extent and the complexity of the specification it is hard to create actually standard compliant process models without tool support. The same problem arises if the standard compliance of given process models should be evaluated by asserting the absence of rule violations. The compliance checks are hampered by the vast amount of constraints to respect as well as by the fact that these constraints are not clearly presented in a dedicated list, but are scattered all over the standard document. The specification also contains several editorial flaws and semantic inconsistencies [2, 7]. Unfortunately, the OMG does neither provide any reference implementations, nor a tool to check the conformance of actual process models. They at least provide normative XSDs, but, as shown in [7], a schema validation is not sufficient to check all constraints, as most constraints are not expressed by the XSD files. Thus, an important prerequisite for actual standard compliance checks is an extensive and clear documentation of all relevant constraints defined in the specification. Such a documentation has been published in [4]. Based on this previous work, we present an approach to automatically assess the standard compliance of BPMN process models using the tool *BPMNspector* in this paper.

The remainder of the paper is structured as follows: First, we briefly discuss related approaches dealing with the analysis of standards compliance or conformance in Section 2. Section 3 demonstrates how standard compliance of process models can be assessed by automated checks using the tool *BPMNspector*. The tool is used to evaluate a set of publicly available process models in Section 4, which gives empirical insights into common problems showing that standard compliance still cannot be assumed in practice. The paper is concluded with a summary and a brief outlook on further research directions in Section 5.

2 Related Work

Various related approaches deal with the standard compliance or conformance of process languages and their implementations in modeling tools and engines.

The OMG itself founded the BPMN *Model Interchange Working Group* (MIWG)[3] to foster the interoperability of modeling tools [1, 12]. However, the approach and focus of the BPMN MIWG is different to ours: They analyze tools based on a set of reference models and check whether the tools are capable of loading, storing and recreating the models in a BPMN compliant serialization [12]. Thus, this work improves process models only indirectly as they target the improvement of modeling tools. Moreover, an important aspect is left out by the BPMN MIWG: They only check whether tools are able to create and deal with *valid* models. It is not investigated whether the tools are able to detect *invalid* models and whether the tools prevent the creation of invalid processes.

Apart from the static standard compliance properties of process models, several works concentrate on behavioral correctness of BPMN processes (e.g. [3,8,16]), and BPMN process engines [5,6]. For example, the absence of deadlocks as well as lack of synchronization in BPMN process models is investigated in [16] using an approach based on Petri nets.

A broader view on the quality of BPMN models is investigated in a recent empirical study in which more than 500 models from industry are analyzed: In [13] not only syntactical rules and advanced structural aspects such as deadlock freedom, but also layout and labeling guidelines have been analyzed, which refers to the *pragmatic aspects* of model quality. In total, the authors claim to have checked "35 well-known BPMN guidelines and correctness rules" [13] extracted from BPMN textbooks. Their results regarding structural aspects indicate that

[3] The project website is `http://www.omgwiki.org/bpmn-miwg/doku.php`

99% of the models are syntactically correct. However, in 22% of all models a deadlock has been found and 42% contain a lack of synchronization (i.e., an unintended multiple execution of process parts). On the downside, only a small subset of all correctness requirements are actually checked, as there exist far more than the checked 35 guidelines and rules [7]. Moreover, the syntactical correctness of BPMN process models can not be taken for granted, as all analyzed models have been created with the same modeling tool, thus not giving a representative insight in the whole state of the art in the BPMN modeling tool market.

Closely related to this work, but dedicated to WS-BPEL [14], is an approach to automatically check WS-BPEL's static analysis rules [9]. We are reusing their proposed language independent API as the basis for the implementation of the tool presented in the next section.

3 Automatic Assessment of Standards Compliance

As shown in [7], the compliance assessment of BPMN process models must be automated to be feasible for the more than 600 necessary constraint checks. An XML schema validation of the process model in the normative BPMN 2.0 serialization format [10] is a good start for such an automatic assessment. However, previous studies have proven that an XSD validation alone is not sufficient as it only detects structural violations, but referencing issues and the more sophisticated constraints are not covered [7]. Thus, we have developed the Java tool *BPMNspector*[4] to check the referential integrity and also most of the more sophisticated constraints listed in [4].

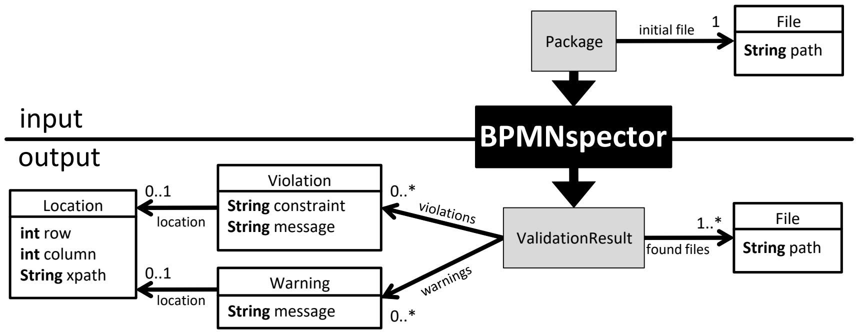


Fig. 1. BPMNspector API Class Diagram (adapted from [9])

BPMNspector enables checks of single BPMN 2.0 process model files as well as whole process model collections - either as a standalone tool or integrated into other tools. The API provided by *BPMNspector* (see Fig. 1) is (process) language independent and has already been used by a tool checking the static analysis rules for WS-BPEL [14] process models [9].

[4] More information can be found on the project web site `http://BPMNspector.org`, the source code is available at GitHub: `https://github.com/uniba-dsg/BPMNspector`

The basic validation workflow is visualized in Fig. 2: After parsing the command line options, the BPMN file, and potentially needed further artifacts (as part of the input package, see Fig. 1) are transformed into an internal `BPMNProcess` representation. During this step the input files are also checked for schema validity. Next, the referential integrity is checked, i.e., do all referenced elements exist and do they have an allowed type. In the final step, the so called *EXT* constraints [4] are evaluated. The *EXT* constraints describe more complex constraints which are not expressed in the normative XSDs. They are defined in a declaritve way using Schematron [11], which is an ISO standard dedicated to define and check constraints for XML-based documents. If a violation of a constraint is found, it is added to the *ValidationResult* which is the output of the tool (see Fig. 1) and basis for the reports created after the execution of all validation steps.

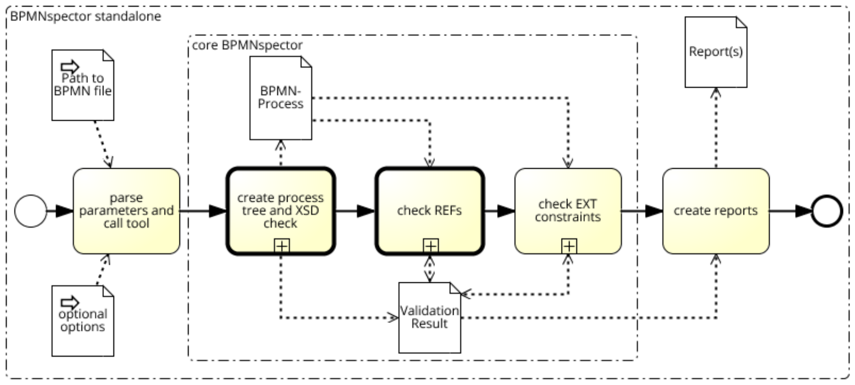


Fig. 2. Basic Workflow of BPMNspector (single file validation)

The reports, either in XML or HTML format, clearly state whether the checked file is valid. For invalid processes each warning, or found constraint violation, is described by a textual explanation and a pointer to the error location.

4 Empirical Insights and Discussion

The tool *BPMNspector* has been used to evaluate publicly available models from different process collections. The first source is the BPMN-by-example document provided by the OMG[5]. These models have been created by BPMN experts as part of standard development to demonstrate the capabilities of the BPMN specification. Although they are officially marked as "non-normative", these models are widely used and referred to both in academia and practice. Thus, they should be valid to avoid confusion of users and tool developers. Another set of process has been created by BPMN MIWG[6] to assess the standard compliance of

[5] see `http://www.omg.org/cgi-bin/doc?dtc/10-06-02`

[6] available at `https://github.com/bpmn-miwg/bpmn-miwg-test-suite/tree/master/Reference`

Table 1. Result Overview: Evaluated Process Collections

description	**models**	**valid**	**invalid**
BPMN-by-example	26	16	10
BPMN MIWG reference models	12	8	4
Collection of QuDiMo Project	32	0	32
totals	**70**	**24**	**46**

modeling tools. As these models are the basis for the evaluation of the standard compliance, they itself must be standard compliant for sensible results. More complex models are included through a process collection of the research project QuDiMo[7].

In total, 70 process models are taken into account, ranging from a very small model containing only 9 process elements, to a complex model containing 364 process elements in an XML file with 2361 LoC. Table 1 provides an overview of the used collections and the number of valid and invalid processes.

The results regarding standard compliance are rather astonishing and a contradiction to the findings of Leopold et al. in [13]: None of the process models provided by QuDiMo is actually completely standard compliant. 27 of the checked 32 models are not even schema valid, mostly because required attributes or sub elements are missing. Furthermore, also the models created by the experts of the BPMN MIWG and the standardization task force contain errors: Whereas only one analyzed model is schema invalid, more than one third of the BPMN-by-example and BPMN MIWG models contain at least one violation of a standard constraint. For example, various BPMN MIWG models do not have all mandatory attributes which "MUST" be present according to the specification, if a `Process` is declared as `executable`.

Most detected constraint violations are related to the wrong usage of `Sequence-Flows`. Some SequenceFlows are completely missing which results in unconnected process elements. Another SequenceFlow related issue, especially occuring in the QuDiMo models, is the connection of two elements from different participants with a SequenceFlow. Instead, these elements must be connected by a Message-Flow [10, Chap. 9.3, p. 111]. Other frequent issues are non-allowed/missing event definitions or missing declarations for executable processes, such as mandatory `ItemDefinitions`. All in all, the 70 processes contain 954 found XSD schema or other constraint violations. And only 24 of 70, i.e., roughly 34%, are completely standard compliant according to the results of our compliance check.

There are various potential reasons for these findings: First of all it would be possible that the tool *BPMNspector* is incorrectly implemented and thus reports issues that are actually non-existent. To minimize this risk we adopted several measures: The implementation is based on [4] which lists the constraints to be respected by valid BPMN process models. All those constraints are di-

[7] The project website `http://pi.informatik.uni-siegen.de/qudimo` provides more information and allows for the download of the process collection.

rectly extracted from the specification and for each constraint a reference to the relevant passage in the specification is given. The source code of *BPMNspector* is thoroughly tested with more than 450 process models we created for testing purposes. Moreover, the list of constraints and the whole source code is open to public scrutiny and we have reviewed both artifacts perpetually within our group to ensure the correctness of our results.

Another potential reason for the compliance issues in the analyzed models might be bad tool support. This is especially relevant for the models published by the QuDiMo project as these models are automatically transformed from another format to the normative BPMN serialization. It is also interesting that the BPMN MIWG references models are accepted as a correct reference by all participants in the working group. In fact, using some tools of participating vendors reveals that they do not complain about any issues. The same applies to the BPMN-by-example models: For example, state-of-the-art modeling tools open the “Noble Prize Process” without reporting any errors and also a manual triggering of the syntax checks does not reveal any problems. In the same file *BPMNspector* reports six violations of two different constraints regarding the invalid usage of non-visible, data related aspects. The inability of detecting constraint violations reveals the already mentioned problem of BPMN MIWG as they solely concentrate on the ability to create valid models, without analyzing their error detection capabilities.

Last but not least, the findings in this paper might indicate that the standard itself should be improved and adjusted. Whereas there are some problems regarding the editorial and also semantical quality of the standard text [2, 7], there also exist clearly and strictly defined constraints which are just ignored by most standard implementers. Especially, the aspects regarding data modeling and execution through Web Services calls do not really reflect the current state of implementation in modeling tools and engines [6].

5 Conclusion and Future Work

In this paper, we have discussed reasons for standard compliance problems and have shown how they can be detected by automated checks using the tool *BPMNspector*. *BPMNspector* does not only perform a schema validation, but is capable of checking most other relevant constraints. Using the tool to test a set of 70 processes revealed that only 24 processes are completely BPMN 2.0 compliant. Thus, we have proven that even basic “syntactic correctness” of BPMN 2.0 process models is not guaranteed, as 40% of all models failed to be schema valid, which clearly contradicts previous results [13].

In future work we try to incorporate existing work for behavioral correctness into *BPMNspector*, as such aspects are even harder to detect in process models [13]. Especially the work of Prinz et al. [16] looks promising to reveal deadlocks and lack of synchronization. Furthermore, we plan to integrate the *BPMNspector* tool into existing BPMN modeling software and BPMN execution engines to increase their capability of detecting invalid process models, which are currently rather weak int this respect [6, 7]. Therefore, this help the users and developers to create valid and standard compliant BPMN process models, thereby mitigating one of the omissions of BPMN MIWG. Moreover, we will analyze more processes from open repositories and process collections to gather further insights in the actual usage of BPMN 2.0 and its frequent problems. This should not only indicate common problems but also provide interesting insights for clarifications, improvements and modifications in further revisions of the BPMN specification.

References

1. F. Bonnet, G. Decker, L. Dugan, M. Kurz, Z. Misiak, and S. Ringuette. Making BPMN a True lingua franca. online on BPMtrends.com, 2014.
2. E. Börger. Approaches to Modeling Business Processes. A Critical Analysis of BPMN, Workflow Patterns and YAWL. *Software & Systems Modeling*, 11(3):305–318, 2012.
3. M. Chinosi and A. Trombetta. Modeling and Validating BPMN Diagrams. In *IEEE Conf. on Commerce and Enterprise Computing (CEC), Vienna, Austria*, pages 353–360. IEEE, 2009.
4. M. Geiger. BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik, no. 92, Otto-Friedrich Universität Bamberg, 2013.
5. M. Geiger, S. Harrer, J. Lenhard, M. Casar, A. Vorndran, and G. Wirtz. BPMN Conformance in Open Source Engines. In *9th IEEE SOSE*, pages 21–30, San Francisco Bay, CA, USA, 2015.
6. M. Geiger, S. Harrer, J. Lenhard, and G. Wirtz. BPMN 2.0: The state of support and implementation. *Future Generation Computer Systems*, 2017. (accepted for publication; available online: http://dx.doi.org/10.1016/j.future.2017.01.006).
7. M. Geiger and G. Wirtz. BPMN 2.0 Serialization - Standard Compliance Issues and Evaluation of Modeling Tools. In *5th Intl. WS on Enterprise Modelling and Information Systems Architectures*, pages 177–190, St. Gallen, Switzerland, 2013.
8. P. V. Gorp and R. Dijkman. A Visual Token-based Formalization of BPMN 2.0 Based on In-place Transformations. *Information and Software Technology*, 55(2):365–394, 2013.
9. S. Harrer, M. Geiger, C. R. Preißinger, D. Bimamisa, S. J. Schuberth, and G. Wirtz. Improving the Static Analysis Conformance of BPEL Engines with BPELlint. In *9th IEEE SOSE*, San Francisco Bay, CA, USA, 2015.
10. ISO/IEC. *ISO/IEC 19510:2013 – Information technology - Object Management Group Business Process Model and Notation*, 2013. v2.0.2.
11. ISO/IEC. *ISO/IEC 19757-3:2016 – Information technology – Document Schema Definition Languages (DSDL) – Part 3: Rule-based validation – Schematron*, 2016.
12. M. Kurz. BPMN model interchange: The quest for interoperability. In *8th Intl. Conf. on Subject-oriented Business Process Management*, pages 1–10. ACM, 2016.
13. H. Leopold, J. Mendling, and O. Günther. Learning from Quality Issues of BPMN Models from Industry. *IEEE Software*, 33(4), 2016.
14. OASIS. *Web Services Business Process Execution Language*, 2007. v2.0.
15. OMG (Object Management Group). *Business Process Model and Notation (BPMN) Version 2.0*, 2011. v2.0.
16. T. M. Prinz, N. Spieß, and W. Amme. A First Step towards a Compiler for Business Processes. In *Compiler Construction - 23rd Intl. Conf., Grenoble, France, April 5-13, 2014. Proceedings*, volume 8409 of *LNCS*, pages 238–243. Springer, 2014.

A Conceptual Framework for Understanding Event Data Quality in Behavior Analysis

Xixi Lu and Dirk Fahland

Eindhoven University of Technology, The Netherlands
{x.lu,d.fahland}@tue.nl

1 Background and Motivation

Process mining aims to derive useful insight for improving business process efficiency and effectiveness. These mining techniques rely heavily on event data, in the form of *event logs*, to provide accurate diagnostic information. The quality of such event data therefore has a large effect on the quality and trustworthiness of the conclusions drawn from the mining analysis and the subsequent business decisions made.

Traditional data quality frameworks focus on identifying *quality dimensions* extensively from a *data perspective* and improving the overall data quality in the long term. While long-term data quality improvement is certainly useful, this may not aid analysts in practice who are often faced with the task of analyzing a given log of lower quality in the short term. As result, when the user conducts a certain analysis (e.g., process discovery), these quality frameworks provide little guidance for assessing or improving the quality of data for the analysis [1, 2, 8]. To the best of our knowledge, only the work in [8] presented event data quality issues as specific patterns reoccurring in logs and discussed their possible effects on mining results from an *analysis perspective*.

In the past few years, we have developed numerous approaches to deal with event logs of low quality, for which no conclusive results are obtained when the user applies existing mining techniques. Three main approaches have emerged: (i) a trace clustering technique based on behavior similarity which allows the user to identify process variants and then explore these variants to discover more precise and conclusive models [5]; (ii) a conformance checking technique using partial order traces and alignments should the ordering of events in a log be untrustworthy [4]; (iii) a label refinement technique in cases where labels of events are imprecise and lead to inconclusive models [6]. However, as each approach is dedicated to tackle a particular event data quality issue from an analysis perspective, an overview for understanding the quality issues is missing.

In this positioning paper, we would like to discuss a conceptual framework to help users understand how these quality issues could be presented and interrelated, how our approaches may be positioned and how future data quality issues may be classified. The conceptual framework[1] is visualized as a table: the columns

[1] The term conceptual framework has taken different definitions in different contexts [7, Chap. 1]. In this paper, we consider a conceptual framework as an analytic tool that helps the user to understand and distinguish different concepts and is easy to remember and apply.

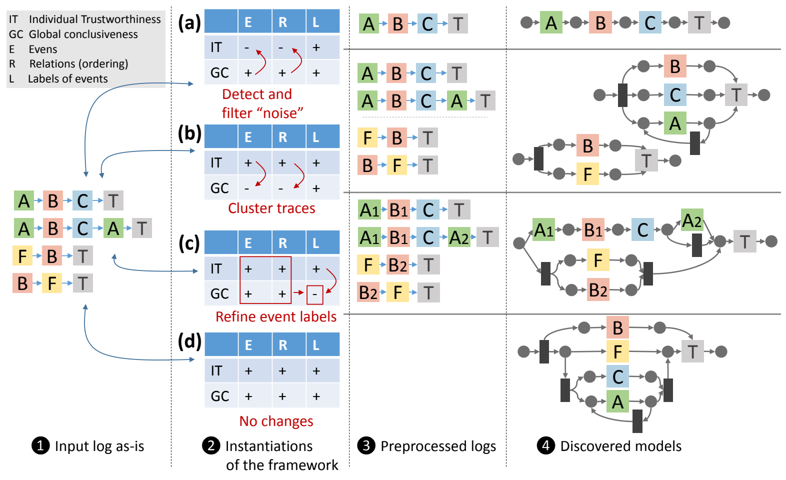


Fig. 1: Four examples of quality issues visualized as possible instantiations of the framework, and the possible preprocessing steps followed.

outline the entities in input data (logs or models) that are relevant for behavior (control-flow) focused analysis; the rows list two dimensions of quality, *individual trustworthiness (IT)* and *global conclusiveness (GC)* which assess the quality of event data from a data perspective and an analysis perspective, respectively. Figure 1 shows four instantiations of the conceptual framework for an event log and is discussed more in depth in Section 2.

2 The Conceptual Framework

In this section, we first explain the framework, its columns, rows and the values assigned to each cell. Secondly, we discuss four prominent cases of event data quality issues and how they are captured by the framework. Finally, we discuss how to extend the framework to capture other cases and conclude the paper.

Our studies into event data quality have shown that there are three entities in event logs, whose quality or trustworthiness have an effect on the results of behavior analysis (e.g., process discovery or compliance checking): (1) quality of *events (E)*, (2) quality of ordering of events, or *relations* among events *(R)*, and (3) quality of *labels* of events *(L)*. These three entities therefore constitutes the columns in the framework.

The quality of each entity is divided into two dimensions: *individual trustworthiness* and *global conclusiveness*. Individual trustworthiness expresses all intrinsic qualities of event data; basically the trust of the user regarding how accurate the event data reflects the real process executions. This quality dimension is similar to *accuracy* or *correctness* dimensions discussed in the literature [1, 3]. However, little research has been conducted into measuring this quality dimension of event data sets. We propose to have three possible values for the individual trustworthiness, as the aim is to allow the user to use the framework with ease and obtain a quick impression of the quality of the data. The three value includes: $+$, which indicates that the user assumes 100% trustworthiness; $-$, which refers to that there are some non-trustworthiness but the majority are trustworthy; $--$, which refers to largely untrustworthy data. For example, the user assigns the individual quality of *events* of a log a $+$ if all events fully reflect the process execution (e.g., fully automated recordings); the user may assign a $-$ if the user thinks there are a few events missing or some duplicated (e.g., when two doctors attended the same consultation for a patient, the consultation might be recorded as two events for the same patient). As another example, the ordering of events in a log might be assigned with $-$ should the user observe that many events happened on the same date and no time is recorded.

Global conclusiveness indicates whether there is a certain path, a certain structure, or a certain pattern that can be observed and is significant, indicating such a pattern is not a random artifact. In other words, this dimension assesses whether there is some behavior, possibly unknown, shared and repeated across a significant number of cases, which implies that there is a particular mechanism or force controlling the flow. Having such a mechanism indicates that future cases would most likely follow this mechanism or pattern. We assign a $+$ if such mechanism is significant enough in a log to be observed and concluded, otherwise a minus $-$. The lack of conclusiveness might indicate the behavior is random or unique, rendering the results of process analysis useless. We acknowledge that conclusiveness is rather difficult to assess or to attribute to only the log or only the model, because conclusiveness may also depend both on the technique applied and on the expectations or understanding of the user of the results. The user may therefore reassess conclusiveness based on the results obtained. Note that there is no trade-off between the two dimensions, a good event data set should be both trustworthy and conclusive in order to perform analysis.

Examples. Figure 1 exemplifies four cases of the framework: the log as-is is shown on the left-hand side; the four tables, one for each case, are shown in the middle of Figure 1; the preprocessed logs and corresponding models are shown on the right-hand side. The first case (a) in Figure 1 is well-known: the user classifies the log as containing some non-trustworthy events (and relations), thus '$-$' for IT of the events and relations. Nevertheless, the log shows the normative behavior (main-flow) rather conclusively, thus '$+$' for GC. Then the analyst may tackle this issue by removing the non-trustworthy cases (or events) and discovering a model from the trustworthy cases. Assume that the variant $\langle A, B, C, T\rangle$ is very frequent and concludes the main behavior, the user can filter out the other cases and discover a simple, sequential model based on this variant.

In contrast, one might classify the same log as globally inconclusive ('$-$' for GC of the events and relations) but individual event as trustworthy ('$+$' for IT), then we have case (b) in Figure 1. To improve the conclusiveness, one might cluster the traces using behavior similarity, since the events and relations among the events are classified as trustworthy. For each cluster, a more precise and conclusive model may be discovered [5]. As in the third case (c) in Figure 1, the user considers some event labels as inconclusive (‘−’ for GC of the labels), while the rest of the log is classified as trustworthy and should contained the main behavior (‘+’ for the rest). Then one may use this information to refine the labels of events by finding similar groups of events that share the same label and the same context. The log with refined labels then yields more conclusive results [6]. Finally, in case of (d) in which all entities are classified as trustworthy and conclusive, one may simply discover a model. If the resulting model is of low quality or inconclusive, this is an indication that the quality of the result does not reflect the (previously assessed) quality of the input. One might revisit the table, reconsider the values assigned to each cell (especially conclusiveness) regarding the applied analysis technique, and improve the quality by preprocessing the log.

Outlook. The columns and the rows of the framework could be extended to tailor the framework towards other analysis. For example, if the user conducts performance analysis in addition to behavior analysis, one could add the entity *timestamps* as a fourth column. Similarly, resources or data attributes could be added as columns. The rows could be extended to other quality dimensions of importance. Interestingly, one may add *model quality* as a row, assessing the quality of the model as an additional quality dimension in the context of a conformance checking analysis. As future work, the applicability of the conceptual framework may be evaluated by conducting empirical studies involving process experts or analysts.

References

1. Bose, R.P.J.C., Mans, R.S., van der Aalst, W.M.P.: Wanna improve process mining results? In: Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on. pp. 127–134. IEEE (2013)
2. Gschwandtner, T., Gärtner, J., Aigner, W., Miksch, S.: A taxonomy of dirty time-oriented data. In: International Conference on Availability, Reliability, and Security. pp. 58–72 (2012)
3. Laranjeiro, N., Soydemir, S.N., Bernardino, J.: A survey on data quality: Classifying poor data. In: 21st IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2015, pp. 179–188 (2015)
4. Lu, X., Fahland, D., van der Aalst, W.M.P.: Conformance checking based on partially ordered event data. In: BPM Workshops 2014. pp. 75–88 (2014)
5. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Detecting deviating behaviors without models. In: BPM Workshops 2015. pp. 126–139. Springer (2015)
6. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Handling duplicated tasks in process discovery by refining event labels. In: BPM 2016. pp. 90–107. Springer (2016)
7. Ravitch, S.M., Riggan, M.: Reason & rigor: How conceptual frameworks guide research. Sage Publications (2016)
8. Suriadi, S., Andrews, R., ter Hofstede, A.H.M., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Inf. Syst. 64, 132–150 (2017)

Towards the Generation of Test Cases for Executable Business Processes from Classification Trees

Thilo Schnelle[1,2], Daniel Lübke[1,3]

[1] FG Software Engineering, Leibniz Universität Hannover, Germany
[2] innoQ Deutschland GmbH, Germany
[3] innoQ Schweiz GmbH, Switzerland

Abstract. Testing executable business processes is essential when developing mission-critical process solutions. However, developers and testers are confronted with two major challenges: Keeping an overview of functional test coverage during test case creation and adopting existing test cases to process changes during maintainance.
Our aim is to develop a generator-based approach that forces the tester to create a classification tree, which keeps track of functional test coverage. Using this classification tree the generator combines test fragments into test cases. Both the classification tree and the code fragments are easier to change than a complete test suite thereby allowing easier test case development and maintance.

1 Introduction

Testing is essential for producing high quality and reliable software [6]. Testing executable business processes that orchestrate services poses special challenges:

1. Processes are usually triggered by messages from external sources and communicate to Web services. External sources can also include user facing applications, like task managers. For sending messages to a process instance and checking messages that are sent by process instances, all partner services need to be mocked. This issue was already addressed by Mayer & Lübke [9] and resulted in the open source tool BPELUnit.
2. Many different flows through the process model, covering both control-flow and data-flow variants, have to be tested. This causes test cases to contain many messages that are sent to the process. Therefore, BPELUnit test suites consist of many lines of code. This amount of code decreases clarity and makes it difficult to keep track of requirement coverage. Finding out whether all requirements from the specification of the process are covered in a test suite with thousands of lines of code poses a significant challenge.
3. A lot of test cases only diverge in later parts of the process. Therefore many messages – especially in the beginning – are repeated often at the start of different test cases. Applying even simple changes to the process can therefore result in changes to every test case.

In this paper we first give an overview over related work done in the field of executable business process testing. Afterwards we propose a generator-based approach that helps testers by reducing the aforementioned problems. We show how the generator is used and the test models are created before we describe two exploratory empirical studies that were conducted to validate and enhance the presented concept before we conclude and give an outlook for future work.

2 Related Work

The use of classification trees was proposed by Grochtmann et al [4]. Their classification-tree editor is used to systematically divide black-box tests up into their requirements. Therefore one subtree per requirement is created and further divided into concrete values that can appear for this requirement. Afterwards they are able to automatically connect these pieces into test cases by picking one value per requirement.

Lübke et al [8] proposed a way to measure test coverage for white-box testing on executable business processes. This approach works by measuring which parts of the code are executed. Therefore it differs from the black-box testing proposed in this paper but can be an additional measure for increasing the quality of test cases for processes.

Another model-based approach by Lübke and van Lessen [7] is based on BPMN-modeled scenarios. This approach also addresses communication with stakeholders and is not focused on testers only. However, no test coverage metrics are included in this approach and also the problem of test case maintainance is not addressed.

3 Classification Trees and Test Meta-Model

Our approach is based on a classification tree as the main management artifact. The structure of this classification tree and the attached technical information is presented in this section. The data structured according to this metamodel is used by a generator in order to create an executable BPELUnit test suite.

The meta-model consists of two parts: First the formalization of the classification tree (figure 1) and second the technical bindings attached to it (figure 2).

The classification tree consists of several categories (first level child nodes) that correspond to different requirements. The categories can have subcategories and values. Values are the leafs of the tree. For example, an image processing software might distinguish between different shapes and colors. Both are independent requirements so they represent categories in the classification tree. Their values are concrete shapes (rectangle, circle, . . .) and colors (red, yellow, . . .). Possible test cases combine these, e.g. a first test case could use red rectangles and a second yellow circles.

For every test case only one value per category may be selected. The goal is to have every value used in at least one test case but to create as few test cases

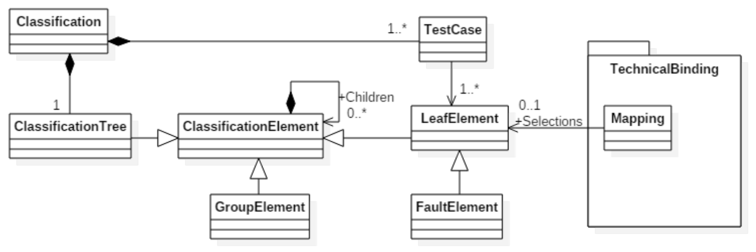


Fig. 1. Meta-Model for Test Classification Trees

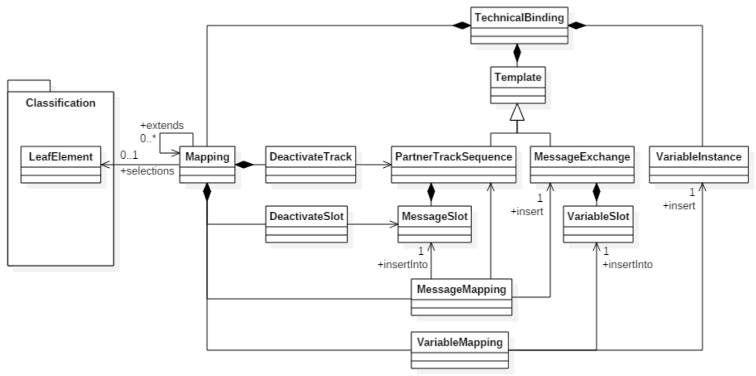


Fig. 2. Meta-Model for Technical Bindings

as necessary. Values that represent errors should always get their own test case as errors may influence the execution of the process drastically.

In the example three test cases are needed to cover all values because the largest category (Shape) contains three values. The maximum number of test cases is 6 (3 shapes times 2 colors).

Although categories should be independent from each other, the pracical use has shown that this strict requirement is very problematic in practice: Often values are independent business requirements but are technically not exclusive. Following the strict theoretical model leads to few categories with many values. In order to allow more readable classification trees, we allow semi-depedent categories and let the testers define conditions that prohibit certain combinations of values. Such extensions have also been proposed by Lehmann & Wegener [5].

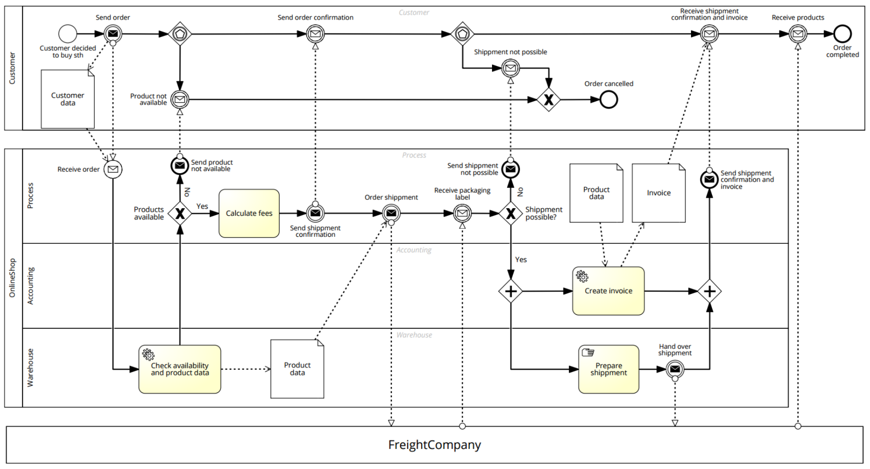


Fig. 3. A simple Example Process for an Online Shop

For every value there is a mapping that defines which test fragments are added to a test case that contains this value. This part is called "technical binding".

Mappings define which messages should be sent at specific points in the execution of the test cases. They also define instances for variable slots. For example in a message there might be a variable slot "purchase date". Possible instances for this slot are concrete dates from which one is chosen. There is also the possibility to deactivate slots for messages and even complete partners of the process, so that any communication to and from a specific service is disabled.

4 Generator Architecture & Example

We implemented a first version of a generator that follows the described meta-model: The classification tree is edited and saved in a spreadsheet and the technical binding information is stored in XML files. The generator reads all files and produces an executable BPELUnit test suite.

In the following, we discuss a simple model of an online shop as a business process related example. Figure 3 shows the corresponding BPMN diagram.

The classification for this online shop is shown in figure 4. There are three independent categories with in total nine values. In contrast to classification trees in "classical" software, the categories can represent process instance data (e.g. message contents) or decisions made in the process (e.g. messages in deferred choices.)

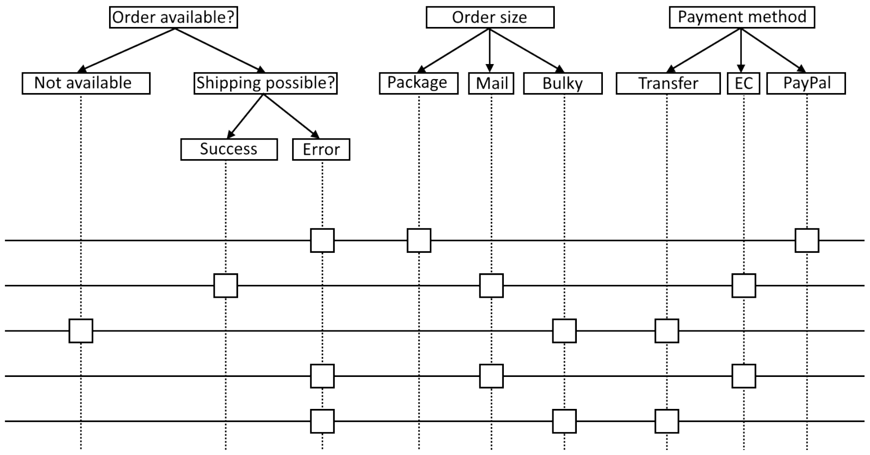


Fig. 4. An Example Classification of an Online Shop

```
<csg:variableDefinitions xmlns:csg="..." xmlns:os="http://www.example.org/OnlineShop/"
xmlns:tes="http://www.bpelunit.org/schema/testSuite">
    <csg:partnerTrackSequence name="ClientPort">
        <csg:messageSlot>OrderAvailable</csg:messageSlot>
        <csg:messageSlot>OrderShipping</csg:messageSlot>
    </csg:partnerTrackSequence>
</csg:variableDefinitions>
```

Fig. 5. Example for a Partner Track Definition

Figure 5 shows the definition of a partner track, which corresponds to a mocked service. For tracks there are slots defined into which message exchanges can be inserted.

These message exchanges are defined like the example in figure 6: The SOAP service, binding and operation are defined like in a normal BPELUnit message exchange but it may additionally contain variable slots. These are named places for the insertion of test case specified data.

This data is defined in variable instances like shown in figure 7. The variable has the same name as the variable slot it can be inserted in.

The connection between the two is made by mappings (see figure 8) and a variable instance is chosen by its name to be inserted into the corresponding variable slots. There are also message exchanges chosen to be inserted into message slots of partner tracks. In addition the figure 8 shows how to deactivate single slots or complete partner tracks if required.

The generator is available as open source at GitHub[4].

[4] https://github.com/ThiloSchnelle/bpelunit-facet-classification-generator

```
<csg:variableDefinitions xmlns:csg="..." xmlns:os="http://www.example.org/OnlineShop/" xmlns:oss=
"www.example.org/OnlineShopSchema/">
    <csg:messageExchange xmlns:tes="http://www.bpelunit.org/schema/testSuite" name="AccountingRequest">
        <tes:receiveSend service="os:Accounting" port="AccountingPort" operation="accountingInformation">
            <tes:receive fault="false" />
            <tes:send fault="false">
                <tes:data>
                    <oss:accountingResponse>
                        <oss:pdfLocation><csg:variableSlot name="InvoiceVar" /></oss:pdfLocation>
                    </oss:accountingResponse>
                </tes:data>
            </tes:send>
        </tes:receiveSend>
    </csg:messageExchange>
</csg:variableDefinitions>
```

Fig. 6. Example of a Message Exchange Definition with a Variable Slot

```
<csg:variableDefinitions xmlns:tes="http://www.bpelunit.org/schema/testSuite"
    xmlns:csg="..." xmlns:oss="www.example.org/OnlineShopSchema/">
    <csg:variableDefinition name="OverallPrice">
        <csg:instance name="Mail">8.99</csg:instance>
        <csg:instance name="Package">93.94</csg:instance>
        <csg:instance name="Bulky">399.99</csg:instance>
    </csg:variableDefinition>
</csg:variableDefinitions>
```

Fig. 7. Example for a Variable Definition

5 Exploratory Empirical Studies

In order to better understand the field of process testing and the implications our approach might have, we conducted two empirical, exploratory studies: A re-implementation of existing test cases of an executable business process taken from an industry project and a small experiment with students.

5.1 Re-Implementation of Industrial Test Cases

The approach was applied to a process of the project Terravis [1], which had a set of unit tests. These tests were re-implemented with the goal of exactly reproducing the existing test suite in order to demonstrate that our approach has the needed expressiveness for defining test cases.

```
<csg:mappings xmlns:csg="...">
    <csg:mapping name="Order available?:Shipping possible?">
        <csg:extends>BaseSelection</csg:extends>
        <csg:variable variableName="ProductsAvailable" variableInstance="Available" />
        <csg:useMessageExchange messageName="NotAvailable" messageSlot="AvailabilityMessageSlot" />
        <csg:deactivateTrack name="AnotherTrack" />
        <csg:deactivateSlot name="SlotX" />
    </csg:mapping>
</csg:mappings>
```

Fig. 8. Example for a Mapping Definition

The size of the original test suite is much larger compared with the size of source files for the generation approach: The original test suite contained 437 declarations of message exchanges that took 3663 lines of code (LOC) while the source files for the generation only declare 53 (12%) message exchanges in 1428 (39%) LOC. The metrics show that the generator approach eliminated much code duplication in the test suite.

During the re-implementation two findings were made:

1. One test case of the existing unit tests used incorrect business data which worked in the process because the data from this service call were not used in this scenario. As a result, the process and the test suite was fixed. Because the generator requires standardized message contents, this defect was found, which is an advantage of our approach.
2. One test case that used anonymized data from production incidents was found. The data was not harmonized with the other test cases. These regression tests should probably not be re-implemented in real-life projects or should be transformed to (re-)use already existing message contents and test data. This also shows that the generator approach probably works best when unified test data is used so that the different binding pieces can be easily joined together and be reused.

5.2 Student Experiment

The experiment took part in a university course. The participants started with no webservice knowledge. They learned to model, implement and test executable business processes. Their task was to test two example processes: One with BPELUnit and one with the presented generator. Afterwards, the students voluntarily answered a questionaire.

Students on average created 2.89 test cases with pure BPELUnit compared to 1.67 test cases with the generator. Some found working with the generator to be faster in the long run and the additional effort to be small. Others found the additional effort big and felt slowed down. The majority judged that the generator is complicated but also increases the overview of the test coverage.

We think that the time and the experience the participants had were not sufficient for the additional complexity introduced by the generator to pay off. This is why developing test cases was slower with the generator. On the other hand an important task of the generator - increasing the overview over test coverage - has been valued by the participants.

From the feedback it also became clear that better tool support is needed. As of now there is no GUI and XML files need to be written manually. This is why we expect better productivity when tooling is in place.

6 Conclusions and Outlook

In this paper we proposed a generator-based approach to testing executable business processes. An initial version of the meta-model and a generator have been developed and has been open sourced. The toolchain has been tried initially in an industrial project.

Moreover, an exploratory student experiment has provided further necessary research and development directions: Especially the missing editor support seems to influence the development speed of the students negatively. Additional tool support needs to be provided and an empirical validation has to be made for it.

Conceptually, the generator-based approach offers the possibility to generate missing test cases by combining different, yet unused combinations of values of the classification tree. One possible algorithm to explore has been presented by Cohen et al [2]. Research with “traditional” software systems has shown that combining all possible values of two attributes with each other yields the highest cost-usage benefits [3]. Further studies need to show whether these findings are also valid for executable business processes.

We will follow up on the open research questions and will especially look at the efficiency (how many defects have been found and comparison between classification tree coverage and code coverage) and would like to join with other academic and industrial partners to further enhance and validate our approach.

References

1. Berli, W., Lübke, D., Möckli, W.: Terravis – large scale business process integration between public and private partners. In: Plödereder, E., Grunske, L., Schneider, E., Ull, D. (eds.) Lecture Notes in Informatics (LNI), Proceedings INFORMATIK 2014. vol. P-232, pp. 1075–1090. Gesellschaft für Informatik e.V., Gesellschaft für Informatik e.V. (2014)
2. Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering 23(7), 437–444 (1997)
3. D. Richard Kuhn, Dolores R. Wallace, A.M.G.J.: Software fault interactions and implications for software testing. IEEE Transactions of Software Engineering 30(6) (2004)
4. Grochtmann, M.: Test case design using classification trees. In: Proceedings of STAR 94. pp. 93–117 (1994)
5. Lehmann, E., Wegener, J.: Test Case Design by Means of the CTE XL. In: Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark (2000)
6. Lübke, D.: Test and Analysis of Service-Oriented Systems, chap. Unit Testing BPEL Compositions. Springer (2007)
7. Lübke, D., van Lessen, T.: Modeling Test Cases in BPMN for Behavior-Driven Development. IEEE Software Sep/Oct 2016, 17–23 (Sep/Oct 2016)
8. Lübke, D., Singer, L., Salnikow, A.: Calculating BPEL Test Coverage through Instrumentation. In: Workshop on Automated Software Testing (AST 2009), ICSE 2009 (2009)
9. Mayer, P., Lübke, D.: Towards a BPEL Unit Testing Framework. In: TAV-WEB ’06: Proceedings of the 2006 Workshop on Testing, Analysis, and Verification of Web Services and Applications, Portland, USA. pp. 33–42. ACM Press, New York, NY, USA (2006)

Events in BPMN: The Racing Events Dilemma

Sankalita Mandal

Hasso Plattner Institute at the University of Potsdam, Germany
sankalita.mandal@hpi.de

Abstract. Today, business process management is a key for companies to represent their operations using business process models. These business processes are executable using process engines. The process engines can produce and consume events for the completion of the processes. However, to receive the external events, we must rely on outer world sources such as a weather API, a traffic agency, an email from a different organization etc. While the digital world makes these message exchanges very convenient, there might still be some latency between the generation of a message and the detection of that message in a receiving process. This latency between the occurrence time and detection time of an event can cause a dilemma of choosing among the alternative paths triggered by racing events and might lead to wrong execution of a process. This problem is investigated in this paper. Also, some solutions are proposed to mitigate the consequences.

1 Introduction

Business processes according to BPMN 2.0 [1] consist of several activities, events and gateways. Events are something that happens in a specific context [2]. These events can be produced or consumed during process execution. Now, in a distributed setup, it can happen that an event has already occurred but the notification of that event occurrence has yet to be received. This may happen due to several manual or system latency. If we think about an application procedure where candidates mail their documents and an administrator enters the information into the system, then it is obvious that the application reception time will be different in the system than the actual reception time via post. In another scenario, when we transfer money via some online payment method, though the payment is done from our side, it may take several minutes or even days to reflect the transferred money in the system of the recipient.

The discrepancy between the occurrence time and the detection time of an event can cause dilemma for choosing among the alternative paths following event occurrences. This paper introduces racing events dilemma in detail and discusses the significance of it, rather the problems that can arise from it. Using two processes for registering to a workshop and paying for the workshop, it is shown that timestamp discrepancy can cause incorrect execution of processes. This may have a negative impact on time, money, user experience or other valuable resources. Possible solutions to mitigate the dilemma are also discussed.

2 Foundation

This section introduces the relevant concepts helpful to follow the rest of the paper. Namely, we talk about business processes consisting of activities and events, especially the racing events and discuss the emerging field of event processing.

2.1 Business Process Model

A business process is a sequence of activities performed in an organizational context where all these activities collectively achieve a business goal [3]. The activities and their coordination are visualized with business process models. Nowadays, BPMN 2.0 is the de facto standard for modeling business processes. Each business process model can be considered as a blueprint for several process instances. An activity model can be instantiated for a set of activity instances.

An activity instance goes through different state transitions as shown in Fig. 1. When a process instance is started, each activity in the process is initialized and is in state *init*. As soon as the incoming flow of an activity is triggered, the instance is in state *ready*. The state changes to *running* when the activity starts execution. Finally, the activity goes to *terminated* state. If the activity instance is *not started* and process instance follows a different path, then it directly goes to *skipped* state. Also, due to the occurrence of an attached boundary event, a running activity instance can be *canceled*.

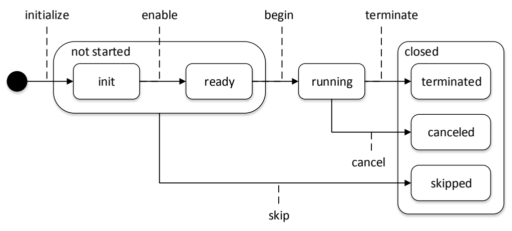


Fig. 1. Activity instance life cycle

For all the activity state changes, the process engine generates transactional events. These are different from BPMN events used in a process model. Let's say t is the function to depict the timestamp of events, be it BPMN event or transactional event. Then to notify the beginning, termination or cancellation of an activity A, the engine logs the events $t_b(\mathrm{A})$, $t_t(\mathrm{A})$, $t_c(\mathrm{A})$, respectively.

Talking about BPMN events, a process model can consist of start events, intermediate events and end events. These events can be of type catching or throwing [1]. For catching events, the control flow enables the event, waits for it to occur and when it occurs, the process consumes it. But there are certain situations where two events race with each other to determine the process flow.

Fig. 2 shows an online booking reservation process for a workshop. After receiving the booking request, the information is processed and the booking is confirmed. Participants can cancel the booking before they receive the confirmation. Once the place is confirmed, booking is not refundable.

To model that, activity `Process Booking` has an interrupting boundary event attached to it. When cancellation is received while this activity is going on, the booking is cancelled and the participant is notified about the cancellation.

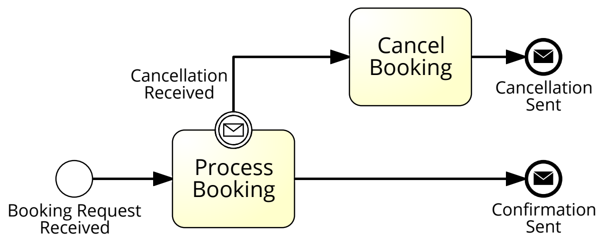


Fig. 2. Interrupting Boundary Event

Here the boundary event `Cancellation Received` is in a race with the event notifying the termination of the activity `Process Booking`.

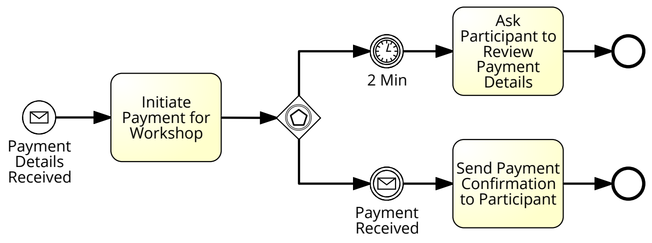

Payment Details Received
Initiate Payment for Workshop
2 Min
Ask Participant to Review Payment Details
Payment Received
Send Payment Confirmation to Participant

Fig. 3. Event-based Gateway

Another example of racing events is an event-based gateway. Fig. 3 shows the payment process for the workshop. Upon receiving the payment information from the participants, the organizers initiate the payment and wait for confirmation. The confirmation is forwarded to the participant once it is received. If confirmation is not received within 2 min then the participant is asked to check the payment information provided earlier. Here the racing events are the message event `Payment Received` and the timer event after the gateway.

2.2 Event Processing

Though BPMN includes a set of different types of events, it does not say much about the source of the events, how they are generated or how they can be correlated. Event processing, on the other hand, explores such concepts [4]. The single or atomic events do not take time to occur, like a weather update from a sensor. Atomic events can be aggregated to generate complex events, e.g., a flight delay message due to five consecutive bad weather updates. Here, the weather updates are collected, matched with the route of the flight and the message is sent to only those passengers who booked this flight.

Event processing platforms like Unicorn [5] are able to perform different operations on event streams. It can receive events, parse them, aggregate them based on predefined transformation rules to create complex events and notify the subscribers about certain events. These notifications can be sent to a person or a system like a process engine that receives this event and reacts on that according to the process specification. The complex events are mapped to the BPMN events included in the executable process models in the process engine. An event e generally has a timestamp attached to it to describe its time of occurrence, identified as t_o(e). The time when the event is detected by a process engine is called time of detection or t_d(e).

3 Problem Statement

If we look back to the example in Fig. 2, it may happen that the moment when the participant pressed the cancellation link was before the `Process Booking` activity finished. Nevertheless, due to server latency the activity terminated before getting notification about the cancellation. In this case, the participants, even if they cancelled the registration timely, do not get the workshop fee back. This problem arises due to the latency between the occurrence time and the detection time of the boundary event. Here, the problem raised as t_o(Cancellation Received) $<$ t_t(Process Booking) $<$ t_d(Cancellation Received).

Considering the example in Fig. 3, there can be a situation where the payment is done within 2 min but the confirmation is not received yet. In that case, the timer fires and the participant is asked to review payment details. Since there was not really any problem with the payment information, it creates confusion. Again, the problem occurs due to the timestamp discrepancy t_o(Payment Received) $<$ t_o(Timer Event) $<$ t_d(Payment Received). In both the cases, with re-evaluation of the situation, the money might be reimbursed or the participant might be informed, respectively. But still it will have negative impacts such as extra time to communicate the problem with the participants, bad user experience and probably some compensation from the organization's side.

One thing to be noted here, the timestamp discrepancy can occur only for those events where the source of the event and the receiver of the event are different. Since timer events are triggered by the engine itself, they are perfectly in sync with the clock that is followed for process execution. Thus, there will be no discrepancy for timer events. So for timer events t_o(e) $=$ t_d(e) will always be true. For other type of events, the timestamp discrepancy can occur due to manual or system delay. If the delay occurs for the start event, the process is instantiated late. In case of an intermediate event, the next activity is started late. But there can be severer impacts due to timestamp discrepancy that leads to a dilemma between the racing events.

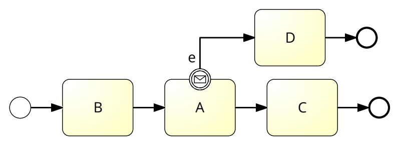


Fig. 4. Problem Statement

Let's look at the problem in general. Fig. 4 shows the boundary event e attached to the activity A. In this case, the two racing events are *e1:* the boundary event *e* and *e2:* the termination of *A*, i.e., t_t(A). Now, the problem occurs when t_o(e1) < t_o(e2) but t_d(e2) < t_d(e1).

According to a correct execution, if e1 occurs before e2, then the path lead by e1 should be chosen. But since the detection time of e2 is earlier than the detection time of e1, the process engine thinks that e2 has happened before e1 and follows the branch lead by e2. Thus, the timestamp discrepancy creates the racing events dilemma which leads to an incorrect process execution.

4 Proposed Solution

Apart from the proactive measures to mitigate the manual delays or increasing system efficiency to communicate faster in a distributed setup, the following measures can be considered to deal with timestamp dilemma.

I: Delay Execution. If we know already that the maximum delay between the occurrence and the detection of the event will be Δ, then we can delay the execution of the process flow accordingly to accommodate the latency. For example, if we know that in case of Fig. 2, $\Delta \leq 10$ sec, then we can set the rule as following. If after t_t(Process Booking) + 10 sec no cancellation is received, then confirmation is sent. Otherwise booking is cancelled. This also has impact on the event subscription since even after the activity terminates, we do not unsubscribe to the boundary event, rather we extend the subscription time by Δ.

II: Perform a posteriori Analysis. The process engine produces event logs where the transactional events such as begin or termination of an activity are listed. Assuming that the BPMN events carry the occurrence timestamp information and the detection timestamp is found in the engine log, we can analyse the traces after a process instance is executed. This tells us if the execution was timestamp correct. To do this analysis, we assume that the systems communicating with each other are using logically synchronised clocks such as Lamport clock [6].

Def: Timestamp Correct. An execution is timestamp correct if for racing events occurrence time and detection time are in same sequence.

- When t_o(e1) < t_o(e2) holds, t_d(e1) < t_d(e2) also holds.
- When t_o(e1) > t_o(e2) holds, t_d(e1) > t_d(e2) also holds.
- For engine generated events, t_o(e) = t_d(e).

According to above definition, the traces 1. and 2. below depict correct execution whereas trace 3. is an incorrect execution for Fig. 4.

1. t_b(B) t_t(B) t_b(A) t_t(A) t_b(C) t_t(C)
2. t_b(B) t_t(B) t_b(A) t_o(e) t_d(e) t_c(A) t_b(D) t_t(D)
3. t_b(B) t_t(B) t_b(A) t_o(e) t_t(A) t_d(e) t_b(C) t_t(C)

The a posteriori analysis will give us insight about the probable event for which there is a discrepancy for most of the instances. Thus, we can try to look for the source of the latency. If it involves manual event generation, we might try to automatize that. If it involves sending events from one platform to another, we might try to increase the efficiency of the platforms such that the delay between occurrence time and detection time of an event is ignorable.

Also, we can calculate the maximum delay for each event from the difference between the occurrence time stated in event timestamp and the detection time logged by process engine. This can be used then to apply solution I.

III: Create Updated Model. Even after detecting the source of latency and taking measures to increase the efficiency it may happen that the discrepancy still occurs. Or it might be the case that the manual or system efficiency cannot be increased very easily. Then we can consider updating the process model to accommodate the latency. In Fig. 3, if the a posteriori analysis suggests that the payment confirmation generally takes longer than expected because of the slow service at the payment gateway, we can set the timer at 5 min instead of 2 min. Thereby, we increase the probability of getting confirmation before the timer fires and decrease number of incorrect execution of the process instances.

5 Related Work

Time is a fundamental concept while working with processes or events. Though BPMN process elements follow sequences based on causality, a sequence in time is also introduced while executing the elements in a certain order. E.g, if activity A must happen before activity B, it implies that the termination of A should be before the beginning of B. But BPMN does not differentiate between the occurrence time and detection time for an event. The existing process engines like Camunda [7] or Chimera [8] react on the events only when they receive it. Therefore, they consider only the detection time of the events.

In a distributed setup, the clocks of different participants should be logically synchronized using Lamport clock [6] or similar algorithm. Though it makes sure that the detection time will always be greater than the occurrence time of the event, it does not solve the problem concerning racing events. The Time-out Reordering mechanism described in [9] stores the events in a queue and delays their detection by a time-out which specifies the maximum delay due to processing of the event. Using this time-out, the events are reordered such that occurrence and detection time are in same order. Woo et. Al [10] have also dealt with the timestamp discrepancy in a similar way. In their frameowrk PTMON, they assume the upper bound of the delay is known and generate RTL formulas to detect probabilistic timing constraints violation. However, none of these methods consider the causal relationship of events with other elements of the process, such as termination of an activity.

The work by Niculae [11] and Eichenberg [12] discuss about different time patterns in workflow management systems where the minimum or maximum time between termination of one activity and beginning of the next activity can be specified. These time patterns can guide us to optimize between the increased waiting time for detecting an event but still starting the next activity timely so that the overall process duration remains acceptable.

6 Conclusion

The BPMN racing events determine the path a process execution should follow. Due to manual delay or processing time needed for communication in a distributed system, there might be discrepancy between the occurrence time of the events and the detection time when the events are received in the process engine. This timestamp discrepancy can cause dilemma between choosing the alternative racing events and even lead to wrong execution of the process. In this work, an investigation into timestamp discrepancy leading to racing events dilemma has been done that shows the possible occurrences of the problem and the impact caused by it. Also, a few proactive and reactive solutions are presented to mitigate the discrepancy that may eventually cause the dilemma. However, further investigation is needed to check the frequency and severance of problems caused by timestamp discrepancy in reality and effectiveness of proposed solutions. Another future direction can be to explore if there exist an alternative to model the processes without racing events so that the problem can be avoided.

References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0/ (January 2011)
2. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications (2010)
3. Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd Edition. Springer (2012)
4. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2010)
5. UNICORN: Complex Event Processing Platform. https://bpt.hpi.uni-potsdam.de/UNICORN/WebHome
6. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM **21**(7) (1978)
7. Camunda: camunda BPM Platform. https://www.camunda.org/
8. Chimera: Case Engine. https://bpt.hpi.uni-potsdam.de/Chimera
9. Hinze, A., Buchmann, A.P.: Principles and applications of distributed event-based systems. Hershey, PA : Information Science Reference (2010)
10. Woo, H., K. Mok, A., Chen, D.: Realizing the potential of monitoring uncertain event streams in real-time embedded applications. IEEE Real-Time and Embedded Technology and Applications (2007)
11. Niculae, C.C.: Time patterns in workflow management systems. Master thesis, Eindhoven University of Technology (2011)
12. Eichenberg, M.: Event-Based Monitoring of Time Constraint Violations. Master thesis, Hasso Plattner Institute (2016)

Tuning Browser-to-Browser Offloading for Heterogeneous Stream Processing Web Applications

Masiar Babazadeh

Faculty of Informatics, University of Lugano (USI), Switzerland
{name.surname@usi.ch}

Abstract. Software that runs on the Cloud may be offloaded to clients to ease the computational effort on the server side, while clients may as well offload computations back to the cloud or to other clients if it becomes too taxing on their machines. In this paper we present how we autonomically deal with browser-to-browser operator offloading in Web Liquid Streams, a stream processing framework that lets developers implement streaming topologies on any Web-enabled device. We show how we first implemented the offloading of streaming operators, and how we subsequently improved our approach. Our experimental results show how the new approach takes advantage of additional resources to reduce the end-to-end message delay and the queue sizes under heavy load.

Keywords: Stream Processing, Peer to Peer Offloading, Autonomic Controller

1 Introduction and Related Work

In recent years the Web has become an important platform to develop any kind of application. With the requirement that applications show results updated in real time, it has become important for developers to use suitable stream processing abstractions and frameworks [1]. In this paper we explore the opportunity to offload part of the computation across different devices on which the stream processing topology is distributed. For example, offloading and migrating part of the computation eases Web servers deployed in the Cloud from computational effort [2], while reducing the response time on clients that can locally perform part of the computation instead of simply rendering the result. Conversely, energy consumption may become a concern and thus the reverse offloading may happen, from clients back to the Cloud [3], or – as we are going to discuss in this paper – to other clients nearby.

In this paper, we take the offloading concept and apply it in a distributed streaming infrastructure [4] where clients and the cloud are tightly coupled to form stream processing topologies built using the Web Liquid Streams (WLS [5]) framework. WLS lets Web developers setup topologies of distributed streaming operators written in JavaScript and run them across a variety of heterogeneous Web-enabled devices.

WLS can be used to develop streaming topologies for data analytics, complex event processing [6], real-time sensor monitoring, and so on. Makers [7] that want to automate their homes and offices with Web-enabled sensors and microcontrollers can use WLS to deploy full JavaScript applicatons across their home server and house sensors without dealing with different programming languages.

Streaming operators are implemented in JavaScript using the WLS primitives, and are deployed by the WLS runtime across the pool of available devices. Users may let the runtime decide where to distribute the operators, or manually constrain where each operator has to run. This is done through a topology description file which is used to deploy and start a topology. The WLS runtime is in charge to bind the operators using the appropriate channels and start the data stream. Depending on the hosting platform, we developed different communication infrastructure. For server-to-server communication we make use of ZeroMQ, for server-to-browser and browser-to-server we use WebSockets, while for browser-to-browser communication we use the recently developed WebRTC.

WLS implements an autonomic controller in charge to increase or decrease parallelism at operator level by adding WebWorkers in bottleneck situations, and removing them when they become idle. At topology level, the controller parallelizes the execution of operators across multiple devices by splitting the operator, or fusing it back together when bottlenecks are solved, depending on the variable data load [8].

In this paper we focus on the controller running on each Web browser and how it can be tuned to make operator offloading decisions based on different policies and threshold settings. This work is based on our previous work on the distributed controller infrastructure [9].

2 Operator Offloading Controller

The Web Liquid Streams controller deals with bottlenecks and failures by migrating streaming operators on available machines, effectively offloading the work from the overloaded machines.

The controller constantly monitors the topology and its operators in order to detect bottlenecks and/or failures and solve them. In particular, it queries the streaming operators as the topology runs, and by keeping into account the length of the queues, the input and output rates, as well as the CPU consumption, it takes decisions to improve the throughput.

We currently have two distinct implementations of the controller infrastructure: one dedicated to Web server and microcontroller operators (Node.JS implementation) and one running on Web browsers. We decided to have two different implementations because of the different kind of environmental performance metrics we can access. In Node.JS our controller has access to many more details regarding the underlying OS, the available memory, CPU utilization, and network bandwidth. For example, to trigger a migration or an operator split in a Web server, the controller needs to check the CPU utilization of the machine and decides to ask for help when it reaches a given specified threshold [9].

The Web browser controller has only access to a subset of the environment metrics, which also heavily depends on the Web browser being used. Thus, the decision policy needs to be adapted accordingly. In a Web browser environment we only know how many CPUs the machine has available through the `window.navigator.hardwareConcurrency` API. We thus decided to use this number as a cap for the number of maximum concurrent WebWorkers on the Web browser.

2.1 First Iteration of the Controller

The first implementation of the Web browser controller was designed to behave very similarly to its Web server counterpart. All the functionalities related to fault tolerance and load balancing were implemented in the same way using a different approach. While the controller cycle was left at 500 milliseconds per cycle, we applied different approaches given the differences in environment.

To compute the CPU usage we relied on the `hardwareConcurrency` API.

$$P(t) > T_{CPU} * \texttt{hardwareConcurrency}$$

When the number of WebWorker threads P on the machine reached the amount of concurrent CPUs T_{CPU} available on the machine (100% CPU capacity), the controller raised a CPU overload flag to the central WLS runtime, which in turn contacted a free host to make it start running a copy of the overloaded operator, thus parallelizing its execution across different machines.

WLS also support flow control to avoid overfilling queues of overloaded operators. This ”slow mode” is also triggered by the controller using a two-threshold rule:

$$\begin{aligned} Q(t) &> Tqh \rightarrow SlowModeON \\ &< Tql \rightarrow SlowModeOFF \end{aligned}$$

The idea behind the slow mode is to slow down the input rate of a given (overloaded) operator to help it dispatch the messages in its queue Q, while increasing the input rate on other instances of said operator. Once the queue is consumed below a given threshold Tql, the controller removes the slow mode, re-enabling the normal stream flow. In [9] we tuned many aspects of the controller, including the slow mode, for three different families of experiments. Results suggested that $Tqh = 20$ messages in the queue were enough to trigger the slow mode, which was released the moment the queue reached $Tql = 10$ or less elements.

2.2 Improving the Controller

By stressing the controller through further experiments we noticed that the metrics given by the `hardwareConcurrency` API were not sufficiently precise to make the controller behave correctly. Very high throughputs and big message sizes in fact showed how the controller took too much time in noticing the bottleneck and asking for help, resulting in very high delays and big queues. We addressed the problem by tuning some controller parameters. We reduced the cycle of the controller from 500 to 300 milliseconds per cycle. We then halved the threshold to trigger the CPU flag to $T_{CPU} = 50\%$.

Finally, we also halved the thresholds to trigger and release the slow mode while maintaining the same formula, obtaining $Tqh = 10$ and $Tql = 5$.

3 Evaluation

To evaluate the performance improvement of the various Web browser controller configurations, we developed a Web cam streaming application. The proposed topology is a simple three-staged pipeline where the producer (first stage) gathers data from the user's Web cam. The filter (second stage) receives the image, runs a face detection algorithm and draws a decoration over the heads of the detected faces, then forwards the result downstream. This stage is intentionally made heavy in terms of CPU time in order to create bottlenecks and stress the controller. The consumer (third stage) shows the image on screen. All the operators run on a Web browser.

The machines used are three MacBook Pros, one with 16GB RAM 2.3 GHz Intel Core i7 (peer 1), one with 4GB RAM 2GHz Intel Core i7 (peer 2), and one with 4GB RAM 2.53GHz Intel Core i5 (peer 3). All the machines run macOS Sierra 10.12 and running Chrome version 45.0.2454.101 (64-bit). We are using an old Chrome version because of recent restrictions related to the use of WebRTC. The WLS server side runs on a 48-core Intel Xeon 2.00GHz processors and 128GB RAM.

For this experiment we deployed the producer and consumer on peer 3, while used peer 1 (P1) and peer 2 (P2) as free machines where the WLS runtime can run and eventually offload the computation of the filter. By default, the WLS runtime picks the strongest machine (peer 1) to run the filter at the beginning, and subsequently offloads the computation on peer 2. We decided to send a total of 6000 messages for this experiment at the rate of 75 milliseconds per message (about 13 messages per second). The size of the Web cam image is fixed to 400x300 pixels, which are converted into messages to be sent, for a total weight of about 800Kb per message. We used the WiFi (averaging around 200Mbit per second) to simulate a normal use case scenario environment.

For this kind of experiment, we decided to focus on the delay as the main metric to measure. In streaming applications that may take into account real-time sensor data, we want to be able to process this data as soon as possible with as little delay as possible. We make use of the queue size as well to show how, by transitioning from one implementation of the controller to another, we are able to keep the queue size small by parallelizing faster and more efficiently.

The results show three different implementations of the controller (C1, C2, C3): the first one and the third one represent the two controller implementations described in Section 2. The second one shows a compromise between the

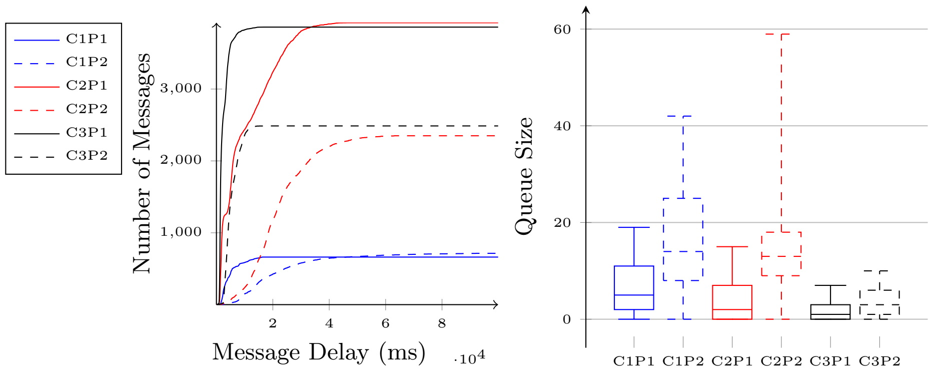


Fig. 1: Message delay and queue size distributions per peer over the three controller configurations.

two, a similar controller with a cycle frequency of 300 milliseconds and with the concurrency check halved with respect to the number of processes on the hosting machine, but with the first slow mode implementation. Table 1 shows the differences in the three configurations.

Figure 1 shows the distribution of the end-to-end message delay of the three configurations as well as the boxplot (mix-max range, mean and quartiles) of the queue size. We cut the delay axis at 100 seconds delay for readability.

The first configuration shows two curves that are slowly growing in terms of number of messages, while keep increasing in delay. This is given by the fact that with our original configuration, under such circumstances, the controller was too slow to raise a CPU flag to the WLS runtime, while the slow mode was triggered too late. The queue boxplots show they were filled and kept being so, inducing message loss and resulting in this small and slowly increasing curve.

The second configuration shows two distinct curves that correctly processed the messages in the topology and processed the vast majority of messages with a delay of 3 to 35 seconds. Both curves follow a similar trend, one being the peer 1 (strongest machine), processing more messages, while the other being on peer 2. By increasing the frequency of the controller cycle and raising a CPU flag sooner, we are able to deal with the increasing queue sizes, keeping them short, and parallelize the work sooner.

A similar trend can be found in the third configuration, where we notice that the majority of messages is executed with less than 10 seconds delay. Both curves

Table 1: Controller configuration parameters

Tuned Parameter	Configuration 1	Configuration 2	Configuration 3
Controller Cycle	500ms	300ms	300ms
T_{CPU}	100%	50%	50%
Tqh	20	20	10
Tql	10	10	5

grow with the same shape, keeping the delay lower than the second configuration. The further improvement in this configuration shows how, by triggering the slow mode earlier, we are able to keep queues even emptier, and thus lowering the delays of the majority of the messages.

4 Conclusions and Future Work

In this paper we have introduced how we approach browser-to-browser operator offloading in the Web Liquid Streams framework. We described our initial control infrastructure and how we improved it to be more responsive in case of increasing workloads. The experimental evaluation shows the benefits of the approach by demonstrating the improvements on the measured end-to-end message delay and operator queue sizes. By increasing even more the workload we may eventually end up filling the queues and all the processors available on every machine. To solve this problem we are implementing support for load shedding [10] that should be triggered by the controller.

References

1. Hochreiner, C., Schulte, S., Dustdar, S., Lecue, F.: Elastic stream processing for distributed environments. IEEE Internet Computing **19**(6) (2015) 54–59
2. Wang, X., Liu, X., Huang, G., Liu, Y.: Appmobicloud: Improving mobile web applications by mobile-cloud convergence. In: Proceedings of the 5th Asia-Pacific Symposium on Internetware. Internetware '13, ACM (2013) 14:1–14:10
3. Banerjee, A., Chen, X., Erman, J., Gopalakrishnan, V., Lee, S., Van Der Merwe, J.: Moca: A lightweight mobile cloud offloading architecture. In: Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture. MobiArch '13, ACM (2013) 11–16
4. Golab, L., Özsu, M.T.: Data Stream Management. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2010)
5. Babazadeh, M., Gallidabino, A., Pautasso, C.: Decentralized stream processing over web-enabled devices. In: 4th European Conference on Service-Oriented and Cloud Computing. Volume 9306., Taormina, Italy, Springer (September 2015) 3–18
6. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44**(3) (June 2012) 15:1–15:62
7. Anderson, C.: Makers : the new industrial revolution. Random House Business Books, London (2012)
8. Hirzel, M., Soulé, R., Schneider, S., Gedik, B., Grimm, R.: A catalog of stream processing optimizations. ACM Comput. Surv. **46**(4) (March 2014) 46:1–46:34
9. Babazadeh, M., Gallidabino, A., Pautasso, C.: Liquid stream processing across web browsers and web servers. In: 15th International Conference on Web Engineering (ICWE 2015), Rotterdam, NL, Springer (June 2015)
10. Gedik, B., Wu, K.L., Yu, P.S., Liu, L.: Adaptive load shedding for windowed stream joins. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management. CIKM '05, New York, NY, USA, ACM (2005)

VISP Testbed - A Toolkit for Modeling and Evaluating Resource Provisioning Algorithms for Stream Processing Applications

Christoph Hochreiner

Distributed Systems Group, TU Wien, Vienna, Austria
c.hochreiner@infosys.tuwien.ac.at

Abstract. Inspired by the current transition towards a data-centric society, more and more researchers investigate cost-efficient resource provisioning algorithms for stream processing applications. While these algorithms already cover a variety of different optimization strategies, they are rarely benchmarked against each other. This makes it almost impossible to compare these different algorithms.
To resolve this problem, we propose the VISP Testbed, a toolkit which eases the development of new provisioning algorithms by providing a runtime for stream processing applications, a library of stream processing topologies, and baseline algorithms to systematically benchmark new algorithms.

Keywords: Data Stream Processing, Testbed, Reproducibility

1 Introduction

Due to the current transition towards a data-centric society, the research area on data stream processing gets more and more traction to tackle the challenges regarding the volume, variety and velocity of unbound streaming data [10] as well as different geographic locations of data sources [5]. A common approach, to implement stream processing applications, is to decompose the data processing into single steps, i.e., operators, and compose topologies as shown in Fig. 1. These topologies can then be enacted by Stream Processing Engines (SPEs), like IBM System S [3], Apache Storm [14], Apache Spark [19], CSA [12] or VISP [7].

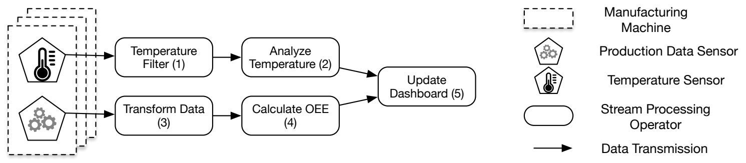


Fig. 1. Stream Processing Topology

The majority of these SPEs is designed, to process a large volumes of data, nevertheless, they often fail to adapt to varying volumes of data. In typical scenarios, like the processing of the sensor data originating from manufacturing machines, as depicted in Fig. 1, the volume of sensor data is often subject to change due to the variable amount of running machines, e.g., due to maintenance downtimes or in error scenarios. This variable volume of data requires different computational resources to process the data in (near) real-time. To comply with the changing resource requirements, it is either possible to over-provision the SPE, i.e., allocate sufficient resources to cover all peaks, or to scale the SPE on demand, i.e., add computational resources only when they are required.

Fig. 2 demonstrates an exemplary scenario for an elastic SPE, which adapts at runtime to the resource requirements of the stream processing application. At the beginning, the incoming data volume is low and it is sufficient to only instantiate one operator of each kind. After some time, the data volume increases and it is required to replicate individual operators to deal with the incoming data volume. As soon as the data volume decreases again, the SPE can release some of the replicated operators while maintaining (near) real-time data processing capabilities. Due to the fact that on-demand resource provisioning is not trivial, several researchers started to investigate different resource provisioning strategies [1,6,9,17].

Although each of these optimization algorithms follows an individual optimization strategy, they have one common challenge when it comes to the evaluation of their algorithm. Up to now, none of the established SPEs provides extensive resource optimization interfaces that can be used to implement and evaluate custom resource provisioning algorithms. While dedicated benchmarking frameworks are available for other areas, like iFogSim [4] for fog environments or Gerbil [15] for semantic entity annotations, there are hardly any projects for resource provisioning algorithms in the stream processing domain. Although there exists several benchmarking projects like streaming-benchmarks[1] or flotilla[2], the scientific community picked up this topic only recently [13]. Nevertheless, to the best of our knowledge there is no framework to benchmark resource provisioning algorithms for SPEs.

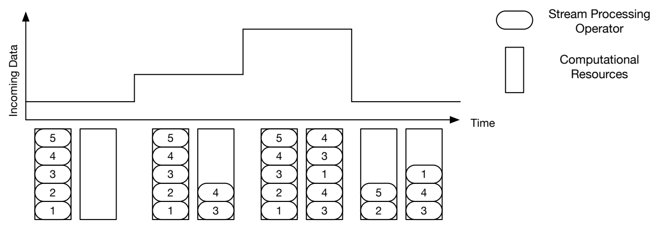


Fig. 2. Resource Requirements for a Deployed Stream Processing Topology

[1] https://github.com/yahoo/streaming-benchmarks
[2] https://github.com/tylertreat/Flotilla

Due to the lack of such a framework, each research group is required to implement an evaluation environment to evaluate their custom algorithm against baseline algorithms instead of state of the art algorithms. To fill this gap, we propose the VISP Testbed as part of our Vienna ecosystem for elastic stream processing, which not only provides a dedicated interface for new resource provisioning algorithms but also different data generation pattern, topologies and baseline algorithms to benchmark new custom algorithms and create reproducible evaluations.

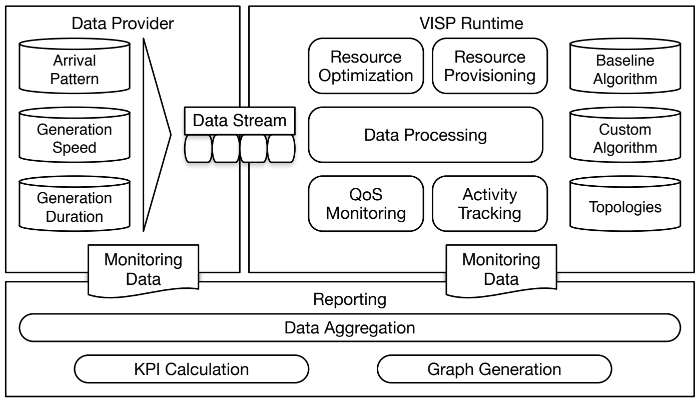


Fig. 3. VISP Testbed

2 System Design

One of the most crucial challenges for realizing a testbed is the data, which is used for the evaluation. This data needs to be recorded from real world systems which makes it very difficult to obtain such data because most data owners do not support the publication of their data. Although there are some data dumps available for scientific purposes, e.g., those published in the course of the DEBS Grand Challenges[3], they are often only suited to evaluate the overall performance of an SPE and not the adoption capabilities of a resource provisioning algorithm. Therefore we decided to implement generic stream processing operators with artificial arrival pattern besides one concrete stream processing topology based on the T-Drive data set [18]. The artificial arrival pattern can be used, to evaluate the adoption capabilities of the resource provisioning algorithms in a clearly defined scenario, before they can be evaluated against real world scenarios without any predefined arrival scenarios. The system design, as shown in Fig 3, consists of three components which are discussed in the remainder of this section.

[3] http://debs.org/?page_id=24

2.1 Data Provider

The Data Provider component[4] is designed to provide a variety of reproducible data streams as input for the VISP Runtime. These data streams can be configured based on their *generation speed*, i.e., how many data items should be generated each second, *generation duration*, i.e., how long should the data be generated and which data generation pattern should be applied. In order to simulate different load scenarios, we have selected four different arrival pattern, based on real world file access pattern [16], as illustrated in Fig 4. These arrival pattern pose different challenges to SPEs as well to its resource provisioning algorithm.

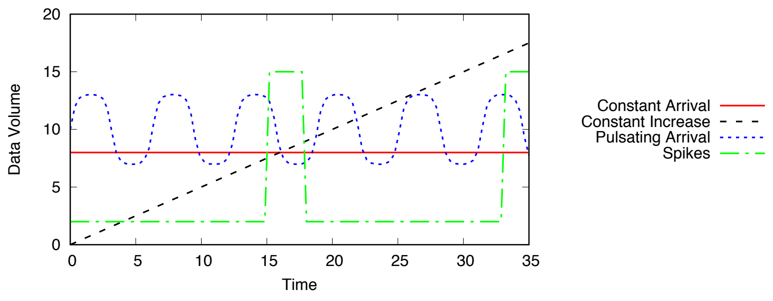


Fig. 4. Arrival Pattern

The simplest arrival pattern is the *constant arrival*, which generates a constant volume of data. This arrival pattern is predestined to test the overall functionality of a resource provisioning algorithm without generating any need for adoption after an initial provisioning.

The *constant increase* pattern is designed to apply stress tests to the SPE and to the resource provisioning algorithm. This pattern is suited to determine the maximal time, for which the resource provisioning algorithm can provide sufficient computational resources for the SPE to comply with given Service Level Agreements (SLAs).

The *pulsating arrival* pattern generates a constantly changing volume of data. This pattern requires the resource provisioning algorithm to constantly update the computational resources for the SPE and is designed to test the adaptation capabilities of the resource provisioning algorithm.

The last arrival pattern is the *spikes* pattern, which has similar properties compared to the constant arrival, apart from short data volume bursts. These short bursts pose high challenges to any resource provisioning algorithm, because they may be prone to allocate a lot of resources for this short time, instead of applying other compensation mechanism like tolerating SLA violations for a short time.

[4] https://github.com/chochreiner/VISP-Dataprovider

2.2 VISP Runtime

The core component of the VISP Testbed is the SPE, i.e., the VISP Runtime[5], as presented in our previous work [6, 7]. The VISP Runtime is designed to process incoming data according to a predefined topology, like the one shown in Fig. 1. Besides the actual data processing of the incoming data stream, the VISP Runtime also takes care of *resource provisioning* for operators and the *resource optimization* thereof based on given algorithms. The *Quality of Service (QoS) monitoring* component of the VISP Runtime monitors different aspects of the data processing, like the processing duration of single data items, the latency between two operators or the individual resource consumption of a single operator. Furthermore, the VISP Runtime also features an *activity tracking* component, which tracks all activities within the VISP Runtime, like upscaling or downscaling of a operators and leasing or releasing new computational resources from a cloud resource provider. These metrics can be either used for the resource optimization but also for the Reporting component, which automatically interprets the evaluation. In order to support future evaluations, we provide a basic library of topologies as well as algorithms.

Evaluation Topologies Our basic library of topologies is motivated by the SAP R/3 reference model [2]. Although the SAP R/3 reference model was originally designed to provide a large variety of business processes for benchmarking purposes, we realized, that today's topologies have similar structures and operations, like data replication (AND-operations) or conditional routes (XOR-operations). Based on this realization, we have selected 10 exemplary topologies based on our previous work [8] to evaluate different scenarios. These exemplary topologies can be equipped with different operators, such as an instant forward operator, a data aggregation operator or a busy-waiting operator to trigger a specific CPU or memory load for each data item.

Baseline Algorithms Currently, the VISP Testbed offers two baseline algorithms, whose configuration, e.g., thresholds, can be parametrized if required.

For the first algorithm, we have selected the *fixed provisioning algorithm*, where the resource provisioning is specified before the evaluation starts and does not change, regardless of the actual data volume. This approach allows to model under-provisioning scenarios, where the SPE has not enough resources at hand to process all data in (near) real-time at peak times as well as over-provisioning scenarios, where a segment of the computational resources is not required most of the time. These two scenarios are suitable to provide a lower baseline for the QoS related attributes, i.e., in an under-provisioning scenario, and the upper baseline for the computational cost, i.e., in an over-provisioning scenario.

For the second baseline algorithm, we have selected a *threshold based* algorithm to adopt the computational resources on-demand. This algorithm replicates operators, when a specific Key Performance Indicator (KPI), e.g., processing

[5] https://github.com/chochreiner/VISP-Runtime

duration, exceeds a threshold and scales it down when these resources are not required anymore. This algorithm provides a more realistic baseline for custom resource provisioning algorithms than the fixed one because it is able to elastically react to varying incoming data volumes.

2.3 Reporting

The final component for the VISP Testbed is the Reporting component, which is currently integrated within the VISP Runtime. This component aggregates the monitoring data from both the Data Provider and the VISP Runtime to automatically generate evaluation reports. For our evaluation reports we distinguish between textual reports, which feature quantitative KPIs, such as the total cost for computational resources, performed provisioning operations or SLA-compliance for the overall data processing, and a graphical representation. For the graphical representation, the reporting component interprets the monitoring time series of the evaluation and generates diagrams with the help of gnuplot[6]. This graphical representations can the be used for more detailed analysis of the resource provisioning algorithm and to identify further optimization potentials.

3 Conclusion

In this paper, we have presented the VISP Testbed, which provides a basic toolkit to conduct reproducible and comparable evaluations with the goal to support the design of future resource provisioning algorithms. Until now, the VISP Testbed only features a small library of topologies and baseline algorithms, but we plan to extend the topology library with concrete topologies from the manufacturing domain [11].

Furthermore, we plan to implement more resource provisioning algorithms based on our ongoing research as well as those from other researchers to provide more competitive baselines. At last, we also want to provide an additional probing component for the VISP testbed, to identify the minimal resource requirements for a specific stream processing operator, which can then be latter used for the resource provisioning algorithms.

Acknowledgments This paper is supported by TU Wien research funds and by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066).

References

1. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator placement for distributed stream processing applications. In: Proc. of the 10th Int. Conf. on Distributed and Event-based Systems (DEBS). pp. 69–80. ACM (2016)

[6] http://gnuplot.sourceforge.net

2. Curran, T.A., Keller, G.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice Hall PTR, Upper Saddle River (1997)
3. Gedik, B., Andrade, H., Wu, K.L., Yu, P.S., Doo, M.: SPADE: The System S Declarative Stream Processing Engine. In: 2008 ACM SIGMOD Int. Conf. on Management of Data. pp. 1123–1134 (2008)
4. Gupta, H., Dastjerdi, A.V., Ghosh, S.K., Buyya, R.: iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. arXiv preprint arXiv:1606.02007 (2016)
5. Hochreiner, C., Schulte, S., Dustdar, S., Lecue, F.: Elastic Stream Processing for Distributed Environments. IEEE Internet Computing 19(6), 54–59 (2015)
6. Hochreiner, C., Vögler, M., Schulte, S., Dustdar, S.: Elastic Stream Processing for the Internet of Things. In: 9th Int. Conf. on Cloud Computing (CLOUD). pp. 100–107. IEEE (2016)
7. Hochreiner, C., Vögler, M., Waibel, P., Dustdar, S.: VISP: An Ecosystem for Elastic Data Stream Processing for the Internet of Things. In: 20th Int. Enterprise Distributed Object Computing Conf. (EDOC). pp. 19–29. IEEE (2016)
8. Hoenisch, P., Schuller, D., Schulte, S., Hochreiner, C., Dustdar, S.: Optimization of complex elastic processes. Trans. on Services Computing 9(5), 700–713 (2016)
9. Lohrmann, B., Janacik, P., Kao, O.: Elastic stream processing with latency guarantees. In: 35th Int. Conf. on Dist. Comp. Systems (ICDCS). pp. 399–410 (2015)
10. McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D., Barton, D.: Big data. The management revolution. Harvard Bus Rev 90(10), 61–67 (2012)
11. Schulte, S., Hoenisch, P., Hochreiner, C., Dustdar, S., Klusch, M., Schuller, D.: Towards process support for cloud manufacturing. In: 18th Int. Enterprise Distributed Object Computing Conf. (EDOC). pp. 142–149. IEEE (2014)
12. Shen, Z., Kumaran, V., Franklin, M.J., Krishnamurthy, S., Bhat, A., Kumar, M., Lerche, R., Macpherson, K.: Csa: Streaming engine for internet of things. Data Engineering pp. 39–50 (2015)
13. Shukla, A., Simmhan, Y.: Benchmarking distributed stream processing platforms for iot applications. In: Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things - 8th TPC Technology Conference, TPCTC. pp. NN–NN (2016)
14. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J.M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., Ryaboy, D.: Storm@twitter. In: 2014 ACM SIGMOD Int. Conf. on Management of Data. pp. 147–156 (2014)
15. Usbeck, R., Röder, M., Ngonga Ngomo, A.C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., et al.: GERBIL: General entity annotator benchmarking framework. In: Proc. of the 24th Int. Conf. on World Wide Web. pp. 1133–1143. ACM (2015)
16. Waibel, P., Hochreiner, C., Schulte, S.: Cost-efficient data redundancy in the cloud. In: 9th International Conference on Service-Oriented Computing and Applications (SOCA). pp. 1–9. IEEE (2016)
17. Xu, L., Peng, B., Gupta, I.: Stela: Enabling stream processing systems to scale-in and scale-out on-demand. In: Int. Conf. on Cloud Engineering (IC2E). IEEE (2016)
18. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: Driving directions based on taxi trajectories. ACM SIGSPATIAL GIS 2010 (2010)
19. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. HotCloud 10, 10–17 (2010)

Closed-Loop Control of 3D Printers via Web Services

Felix W. Baumann[1] and Dieter Roller[1]

University of Stuttgart, Stuttgart BW 70569, Germany,
felix.baumann@informatik.uni-stuttgart.de

Abstract. In this work, we present a method to directly control an existing 3D printer with a retrofittable device and communication to and from an online service. The control is exerted utilizing calls to a RESTful API that provides functionality for an online 3D printing service. The direct control of the hardware aims to alleviate problems with the printing process and a fine grained control for a high-quality printed object. The system relies on online data acquisition for which this work provides an implementation based on previous work and extended to a custom made sensor node.

Keywords: 3D Printing, Additive Manufacturing, Online Service, Hardware Control

1 Introduction

3D printing or synonymously Additive Manufacturing (AM) describes the creation of physical objects from digital models [9]. The creation of the objects is tool-free and based on the geometrical information from the digital models. 3D printed objects are created using a range of different technologies and materials [12, 15]. For consumer-grade 3D printers the material commonly used is either Acrylonitrile butadiene styrene (ABS) or polylactic acid (PLA), both are thermoplastics, that are heated above the glass transition temperature, so that the plastic is soft and extrudable. The heated thermoplastic is extruded through a nozzle onto the print-bed or a previous layer of the object, where it solidifies by cooling. This technology is called Fused Deposition Modeling (FDM) or Fused Filament Fabrication (FFF) [7]. With this technology, as with most other technologies, the object is created in a layer-wise fashion, with the material being deposited at specific places that are defined in the machine code. Other technologies include the sintering of metal powder (Direct Laser Sintering) using a laser beam for the creation of metal or ceramic objects. 3D printers for the consumer market are open-loop controlled which means that there is no feedback within the machine on the current operating status. This lack of direct control can lead to errors and misprints caused by external or internal factors. As an example, vibration from the machine itself or from external sources, can cause the object to detach, causing a failed print. In an error scenario, the 3D printer is occupied for the duration of the 3D print, without producing the requested resulting object, thus blocking the 3D printer itself and wasting material, both the filament and the 3D printer internal components that are specified as usable for a limited and specific amoutn of time. A number of error states are characterised by a build up phase where the fault is not permanent but will be if unadjusted. With this research we include sensors for vibration and acoustics in order to detect such faulty states early. The system we propose is backed by an online 3D printing service, based on previously published works, that features a RESTful API. It is elaborated on the timing requirements as well as required latency for the early detection of faulty states, and the adaption and correction of machine instructions. This work is describing the service requirements for an extension of a service to extend and interface with real-world devices, thus forming a Cyber-physical system (CPS). The scope of this work is to define the requirements and architectural decisions for such a CPS to control an existing 3D printer via an external, remote service in a closed-loop control manner. Furthermore, this proposed system is not limited to 3D printing but can be utilised in the remote monitoring and control of other hardware, such as CNC or milling machines, where sensorial output can be acquired and the control is possible through computerised systems. Remote monitoring and control capabilities are also integral concepts for Industry 4.0 scenerios. In the current state, no internal models of the 3D printing process exist for this work. The intention is to compensate for errors that are resulting from exessive vibration of the 3D printer, thus allowing for better quality results and reduced power motors thus saving money and extending the service life time of 3D printers.

2 Related Work

The work by Lotrakul et al. [10] explores the possibility of state and fault detection in AM utilizing acoustic emissions. The authors were able to both detect filament supply loss and nozzle clogging in their experiment. The information available through acoustic analysis is further presented in the work by Song et al. [11] where the authors describe a side-channel attack on the intellectual property during the printing process.

In contrast to the proposed closed-loop system by Weiss in his master thesis [13] and the corresponding article [14] where the control is directly interfacing with the motor control, this system will not alter the motor control but will interface through the exposed firmware of the 3D printer that accepts machine commands through a USB-serial interface. The necessity for an improved process supervision and control is presented in the work by Blandon et al. [6]. In this work the authors presented a list of potential challenges and problems that can occur during the 3D printing process.

The authors Faes et al. [8] present an alternative monitoring strategy for AM utilizing laser scanning. Their work is focussed on the monitoring of process properties, whereas the following work by Xiong et al. [16] is focussed on the control strategy.

In Xiong et al. [16] the authors present a closed-loop control system for wire and arc AM (WAAM), based on an artificial neural network.

Existing proposals for direct-control of 3D printing hardware is focused on the direct integration of closed-loop control systems within the 3D printer, respectively the firmware. The system proposed in this work, on the other hand, utilises an external service for the control component. This externalisation brings benefits such as flexibility and extensibility at the drawback of latency and the strict requirement of availability of or connectability to remote components.

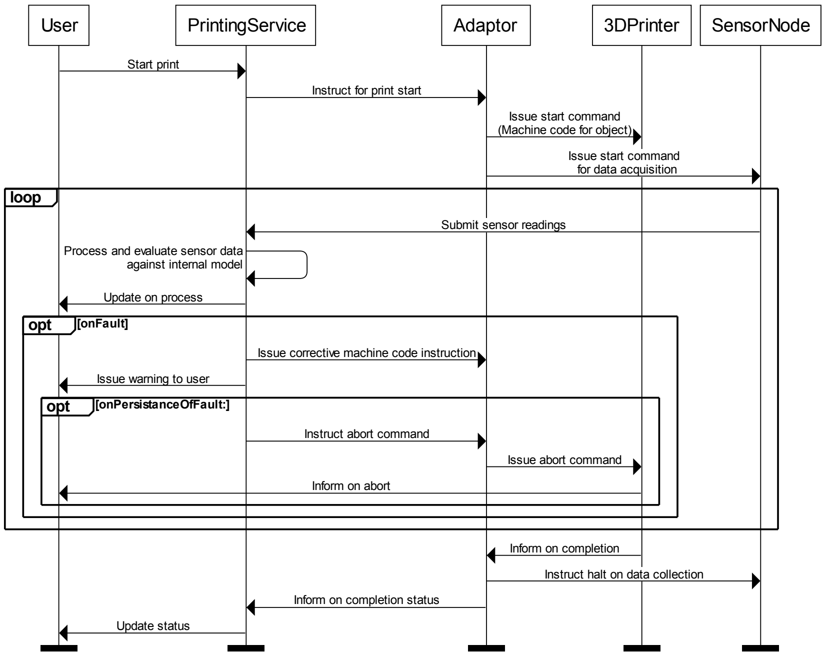
3 Proposal

The system described within this work is based on three components: 1. Sensor Node 2. 3D Printer Adaptor and 3. Printing Service. The sensor nodes are based on custom built PCBs (Printed circuit board) with a STM32F072C8[1] MCU (Microcontroller unit), a MPU9250[2] IMU (Inertial measurement unit) and a SPU0410HR5H-PB[3] microphone. The sensor nodes are extensions to previous works presented in [2, 4]. The algorithms for the analysis and evaluation of the sensor data is implemented in the cloud printing service, presented in [1, 5], as a service component that is exposed through the service API and offers RESTful endpoints to the internet. See Figure 1 for the communication flow between the individual components during printing. In this figure, the adaptor and the *PrintingService* are divided by the Internet, and the User is also divided by the Internet from the *PrintingService*. The *SensorNode* is directly attached to a 3D printer and acquires data throughout the 3D printing process. A configurable number of *SensorNodes* is attached to and controlled by a computer system titled *Adaptor* which communicates with the 3D printer and the *PrintingService*. The *PrintingService* is equipped with a model of the expected behaviour of the 3D printing process against which the acquired sensor data is checked. These models are specific to a 3D printer and are to be generated through observation or simulation.

Preliminary calculations on the speed and latency requirements are based on the experiment conducted in Baumann et al. [3]. In this experiment, 121 specimens were printed using an FDM 3D printer (*Makerbot Replicator 2X*) and the executions were captured and analysed. The objects are of different shape and size. From the actual printing time, it was found that the average movements involving the X-axis were 7.25 per second and 8.57 per second for the Y-axis. The average distance travelled by the printhead involving the X-axis is 23.42 $\frac{mm}{s}$ and 21.95 $^{mm}/_{s}$ for movements involving the Y-axis. From these numbers, it can be calculated that the average time for any instruction involving the X-axis is 0.16 s and 0.14 s respectively. In order to provide rapid interaction with the printing process the lower number of 0.14 s is the hard limit

[1] `http://www.st.com/en/microcontrollers/stm32f072c8.html`
[2] `https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/`
[3] `http://www.knowles.com/eng/content/download/5754/91789/version/3/file/SPU0410HR5H-PB+revH.PDF`

Fig. 1. Component Communication Flow


User
PrintingService
Adaptor
3DPrinter
SensorNode
Start print
Instruct for print start
Issue start command (Machine code for object)
Issue start command for data acquisition
loop
Submit sensor readings
Process and evaluate sensor data against internal model
Update on process
opt [onFault]
Issue corrective machine code instruction
Issue warning to user
opt [onPersistanceOfFault:]
Instruct abort command
Issue abort command
Inform on abort
Inform on completion
Instruct halt on data collection
Inform on completion status
Update status

in which the sensor data acquisition, data transmission, analysis, correction and instruction re-transmission must occur, i. e., one round-trip. This is estimated so that only one instruction will be performed without correction thus limiting potential damage and increasing the chance to recover or compensate.

With an assumed sensor data acquisition frequency of 400 Hz the data transmission must occur in bundles of less than 56 measurements to stay under the hard threshold given above. As the transmission in bundles versus single data transmission is a tradeoff between latency and transmission volume the following bundle sizes are tested: 10, 20, 30 and 40 measurements per data submission. An experiment is conducted in two locations for two application scenarios. The first scenario is the deployment within a university and the second is the deployment with a customer connected to the Internet using a DSL (Digital Subscriber Line) connection. From preliminary experiments a clear discrepancy between the latency of these two locations is evident, see Table 1.

In the following Table 1, the latency to a number of Internet hosts is listed. *Hosts A* through *C* are on the local network α and *hosts D* through *F* on the local network β. The roundtrip time listed is achieved for ICMP requests to the respective hosts and used to demonstrate the potential problems for the proposed service, based on the available Internet connection. The hosts tested are selected arbitrarilly and can be regarded as prototypical connections to and from universities, inter-corporation connection, local and overseas connections.

Table 1. Roundtrip Times in ms from Locations Connected to the Internet using a DSL and a 10G Ethernet connection

	DSL/Network α			10/100 GE/Network β		
Host		rtt			rtt	
	min	**avg**	**max**	**min**	**avg**	**max**
heise.de	19.975	20.665	21.692	4.335	4.455	4.691
uni-stuttgart.de	31.235	31.912	32.759	0.343	0.396	0.436
fh-esslingen.de	31.75	32.34	33.116	0.734	0.877	1.083
kit.edu	41.003	42.113	51.259	2.563	2.711	4.425
Host A	1.324	1.545	1.930			
Host B	4.081	5.134	14.389			
Host C	2.100	2.725	5.171			
1und1.de	19.974	20.42	21.281	3.831	3.868	3.901
acm.org	106.608	108.572	117.772	84.561	84.959	87.13
Host D				0.189	0.229	0.314
Host E				0.130	0.251	0.400
Host F				0.221	0.353	0.559
mashup.inf.unisi.ch	24.107	25.033	77.941	6.874	7.022	7.210
Average		29.0459			10.5121	

The required data rates vary for the sensornode configurations and the amount of sensornodes. Exemplary it can be defined, that the throughput must at least be sufficient to stream from three sensornodes with three sensors each (acceleration, temperature, gyroscope) at a data acquisition rate of 400 Hz per sensor. For this scenario, the raw data to be transported is 49.2 $^{Kib}/_{s}$, as the acceleration and gyroscope report data on three axis each and each sensor value is of 16 bit size. This throughput is calculated without protocol overhead and assumed package loss.

4 Benefits and Drawbacks

With the proposed system, the extension of the closed-loop control capability on new 3D printers is enabled without the requirement to change the firmware of the respective 3D printer or the alterations in hardware besides the addition of the sensornodes. Different 3D printing mode models can be implemented in the service and selected by the user as per his requirement. The modes can differ in the fault tolerance or achievable 3D print quality. As a drawback of the proposed system, the constant and uninterupted Internet connectivity is stated. This connection must be capable to provide high-throughput and low latency as per the requirements stated. These requirements can vary for sensornode configurations.

5 Conclusion

This work analyses the time available for each component respective transmission sequence of information so that the hard threshold is not exceeded. With this approach provided, it is possible to provide closed-loop control of existing 3D printers, by the utilisation of a remote service. Potential benefits, such as extensibility and flexibility for the addition of new 3D printers, is discussed as well as the drawbacks of requiring constant and uninterupted Internet connectitvity. The work presented in this paper is still in progress and requires the extended implementation and evaluation of the proposed service and adaptor to generate the required models of the 3D printing process per individual 3D printer.

References

1. Baumann, F., Eichhoff, J., Roller, D.: Collaborative Cloud Printing Service, pp. 77–85. Springer International Publishing (2016), `http://dx.doi.org/10.1007/978-3-319-46771-9_10`
2. Baumann, F., Schön, M., Eichhoff, J., Roller, D.: Concept Development of a Sensor Array for 3D Printer. In: Procedia CIRP, 3rd ICRM 2016 International Conference on Ramp-Up Management. vol. 51, pp. 24–31 (2016), `http://dx.doi.org/10.1016/j.procir.2016.05.041`
3. Baumann, F., Straßer, S., Roller, D.: Free complexity in additive manufacturing?! a study in fused deposition modeling. 3D-Printed Materials and Systems (2017), in preparation
4. Baumann, F.W., Eichhoff, J.R., Roller, D., Schön, M.: Sensors on 3D Printers for Cloud Printing Service. In: Proceedings of the 2016 International Conference on Advanced Material Science and Mechanical Engineering (March 2016), `http://www.amsme2016.org/sub.htm`
5. Baumann, F.W., Kopp, O., Roller, D.: Abstract api for 3d printing hardware and software resources. International Journal of Advanced Manufacturing Technology (2016), submitted - Under Review
6. Blandon, S., Amaya, J.C., Rojas, A.J.: Development of a 3d printer and a supervision system towards the improvement of physical properties and surface finish of the printed parts. In: 2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC). pp. 1–7 (10 2015), `http://dx.doi.org/10.1109/CCAC.2015.7345179`
7. Bonten, C.: Kunststofftechnik. Carl Hanser Verlag GmbH & Co. KG (2014), `http://www.hanser-fachbuch.de/buch/Kunststofftechnik/9783446440937`
8. Faes, M., Abbeloos, W., Vogeler, F., Valkenaers, H., Coppens, K., Goedemé, T., Ferraris, E.: Process monitoring of extrusion based 3d printing via laser scanning. In: Proceedings of the International Conference on Polymers and Moulds Innovations (PMI). vol. 6, pp. 363–367 (2014), `https://lirias.kuleuven.be/handle/123456789/451935`
9. Gibson, I., Rosen, D., Stucker, B.: Additive Manufacturing Technologies - 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing. Springer New York, 2 edn. (2015), `http://dx.doi.org/10.1007/978-1-4939-2113-3`
10. Lotrakul, P., San-Um, W., Takahashi, M.: The Monitoring of Three-Dimensional Printer Filament Feeding Process Using an Acoustic Emission Sensor, pp. 499–511. Springer Singapore, Singapore (2017), `http://dx.doi.org/10.1007/978-981-10-0471-1_34`

11. Song, C., Lin, F., Ba, Z., Ren, K., Zhou, C., Xu, W.: My Smartphone Knows What You Print: Exploring Smartphone-based Side-channel Attacks Against 3D Printers. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 895–907. CCS '16, ACM, New York, NY, USA (2016), http://dx.doi.org/10.1145/2976749.2978300
12. Turner, B.N., Strong, R., Gold, S.A.: A review of melt extrusion additive manufacturing processes: I. Process design and modeling. Rapid Prototyping Journal 20(3), 192–204 (2014), http://dx.doi.org/10.1108/RPJ-01-2013-0012
13. Weiss, B.: Closed-Loop Control of a 3D Printer Gantry. Master's thesis, University of Washington (10 2014), http://hdl.handle.net/1773/26048
14. Weiss, B., Storti, D.W., Ganter, M.A.: Low-cost closed-loop control of a 3d printer gantry. Rapid Prototyping Journal 21(5), 482–490 (2015), http://dx.doi.org/10.1108/RPJ-09-2014-0108
15. Wong, K.V., Hernandez, A.: A review of additive manufacturing. ISRN Mechanical Engineering 2012, 1–10 (2012), http://dx.doi.org/10.5402/2012/208760
16. Xiong, J., Yin, Z., Zhang, W.: Closed-loop control of variable layer width for thin-walled parts in wire and arc additive manufacturing. Journal of Materials Processing Technology 233, 100–106 (2016), http://dx.doi.org/10.1016/j.jmatprotec.2016.02.021

Highly Scalable and Flexible Model for Effective Aggregation of Context-based Data in Generic IIoT Scenarios

Simon Duque Antón, Daniel Fraunholz, Janis Zemitis, Frederic Pohl, and Hans Dieter Schotten

German Research Center for Artificial Intelligence
Intelligent Networks Research Group
{firstname.lastname}@dfki.de

Abstract. Interconnectivity of production machines is a key feature of the Industrial Internet of Things (IIoT). This feature allows for many advantages in producing. Configuration and maintenance gets easier, as access to the given production unit is not necessarily coupled to physical presence. Customized production of goods is easily possible, reducing production times and increasing throughput. There are, however, also dangers to the increasing talkativeness of industrial production machines. The more open a system is, the more points of entry for an attacker exist. Furthermore, the amount of data a production site also increases rapidly due to the integrated intelligence and interconnectivity. To keep track of this data in order to detect attacks and errors in the production site, it is necessary to smartly aggregate and evaluate the data. In this paper, we present a new approach for collecting, aggregating and analyzing data from different sources and on three different levels of abstraction. Our model is event-centric, considering every occurrence of information inside the system as an event. In the lowest level of abstraction, singular packets are collected, correlated with log-entries and analyzed. On the highest level of abstraction, networks are pictured as a connectivity graph, enriched with information about host-based activities. Furthermore, we describe our work in progress of evaluating our aggregation model on two different system settings. In the first scenario, we verify the usability of our model in a remote maintenance application. In the second scenario, we evaluate our model in the context of network sniffing and correlation with log-files. First results show that our model is a promising solution to cope with increasing amounts of data and to correlate information from different types of sources.

Keywords: Data Aggregation, Industrial Internet of Things, IT-Security, Big Data, Complex Event Processing

1 Introduction

The fourth industrial revolution is a driver that leads to increased interconnectivity of production assets. Machines and goods are equipped with intelligence as well as communication abilities to negotiate processing steps. As a result, a lot of communication takes place, within a production site but also across the boundaries of several facilities. For several reasons, such as security and intrusion detection, pre-emptive maintenance, quality management and error detection, it is necessary to monitor this communication data. The data generated by smart devices also contains relevant information about conditions of entities. Furthermore, the information of controlling entities such as Manufacturing Execution Systems (MES), Enterprise Resource Planning (ERP) or maintenance ticketing systems can be used to validate the soundness of device behaviour. In this paper, a holistic approach to collect and connect this data is presented. In our approach, every information is derived from time-discrete events. It therefore contributes to the field of Complex Event Processing (CEP) in the domain of Industrial Internet of Things. We describe a model to classify the data that can be extracted and a way to connect data from different sources. This is especially useful to take the context of events into consideration and to semantically assess them. This paper is organized as follows. In chapter II, the state of the art of monitoring and aggregation of data as well as context-aware data analytics is described. An exhaustive specification of our data aggregation model is presented in chapter III. In chapter IV, the application of our model to two use cases is evaluated. A conclusion is drawn in chapter VI, as well as an outlook on the continuation of our work.

2 Related Work

Event processing has been brought up in the 1950's in the context of discrete event simulation [12]. Since then, a lot of development has taken place. Nowadays, because of the rising amount of data and the increasing complexity of information, databases and sophisticated queries are employed [5, 16]. This field is called "CEP" and applied on many domains, often on business and finance intelligence [4, 7], but also on education [6] or automation purposes [14, 15]. For network management and IT security, similar concepts are used under the name of Event Correlation [8,10,11]. The methods of CEP are used to correlate, aggregate and access information in order to gain intelligence that is only available due to the combination of multiple events [13]. Most CEP systems are based on data bases and tailored query languages, for example SASE [9], but many others as well [5]. One core idea of CEP is gaining and storing only relevant information from a wide range of sources. This is called "aggregation". One of the current issues of CEP is the horizontal correlation of events [10,14]. Correlation is used to describe relations of cause and effect. This is a non-trivial topic and is heavily domain dependant task [7,10]. Our model proposes a concept for this issue in the domain of IIoT.

3 Concept of the Aggregation Model

In this section, the concept of the aggregation model is described. It consists of three vertical and horizontal levels each, as can be seen in table 1. The vertical levels depict the layer of abstraction. The lowest level in the table corresponds to the highest resolution, while the highest level corresponds to the most abstract view. To ascend from a level to another, data has to be summarized. The horizontal levels describe the relation between different sources of data. Each column represents a different kind of data source that can be used to get a specific kind of data.

Table 1. Aggregation Model

	Cause	Traffic	Effect
Level 1	e.g. Configuration entries	e.g. Network Packets	e.g. Log-entries
Level 2	e.g. Application Info	e.g. Network Flow	e.g. Log-files
Level 3	e.g. Data base	e.g. Flow Graph	e.g. Log file collection

The leftmost column describes the cause of an event. It is the command or the action on a system that triggers network traffic and an action on a remote system. In the middle column, the network traffic of an event is depicted. The rightmost column describes the effect of an event. It is the action on a remote machine, triggered by some action or the activity a PLC executes due to a change of parameters. We assume a network-centric approach, meaning that only actions that spread within the network are considered relevant. This is because of the increasing importance of interconnectivity and the inherent effects of it on computing. Actions on a single host can translated to information of effect and cause, without any network traffic. This case, however, is not the goal of our model. Those kinds of data need to be correlated in order to gain intelligence on the system that is represented by the model.

The layers are introduced to encode the resolution that is used. On the lowest level, Level 1 in table 1, singular network packets are collected and analyzed. They are correlated with singular log-entries on both cause and effect side. The idea is that a single IP-packet is the smallest unit worth analyzing. On both sides of the communication, actions can be determined that either result in or were caused by this traffic, such as log entries or syscalls. This information can be aggregated and taken to a higher level of abstraction, Level 2 in table 1. The network packets can be aggregated to flows. Flows only contain source and destination as well as duration, number of packets some more meta information. The log-entries on the hosts involved can be aggregated to log-traces, as can the syscalls. On the machine labeled as the cause, application data can be taken into account, depending on the application to gather intelligence on the context of the event. On the highest level of abstraction, Level 3 in table 1, the whole network is taken into consideration. The network-flows are aggregated to network topology graphs.

The log- and syscall traces as well as the application information are stored in a compressed form as node-information and correlated with the corresponding edges in the graph. This allows for a compact, yet complete overview of the network.

4 Use Cases and Evaluation

In this chapter, two use cases of our model are depicted. The first use case is a remote maintenance scenario. In the context of the second use case, modbus-based network traffic is correlated with logs gathered from PLCs.

4.1 Remote Maintenance

In increasingly interconnected production networks, secure remote maintenance gains importance. We apply our model to an architecture for secure session-based remote administration and maintenance that suits the requirements of segmented and firewalled production network environments. The architecture is depicted in fig. 1[1]. It is divided by means of the zones A, B with limited, i.e. firewalled, access to zone C, thus without the ability to setup zone-spanning peer-to-peer connections. The cloud platform takes the role of a relay agent, forwarding legitimate traffic between entities from zone A to zone B and vice versa.

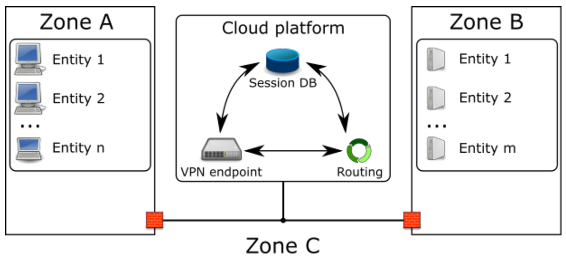


Fig. 1. Architectural Overview Remote Maintenance

The cloud platform consists of three core components, each of them contributing information to the aggregation model. The session database holds information about scheduled, ongoing as well as terminated remote administration sessions. It is queried by the VPN endpoint in order to additionally authenticate connecting entities, as well as the routing engine to setup forwarding tables according to session properties. Any traffic related to remote administration passes the VPN endpoint and can easily be extracted for aggregation. Additional session state information can be obtained by the routing engine. We further note that communication between the core components may also be subject to aggregation.

[1] The icons have been taken from [3]

4.2 Log and traffic correlation

PLCs are computation units in indutrial networks that control production processes. They can be configured via network access. In order to gain intelligence, we propose to collect network traffic data as well as host-based log information on the PLC. The schematic architecture that helps us gather the information necessary is pictured in fig. 2.

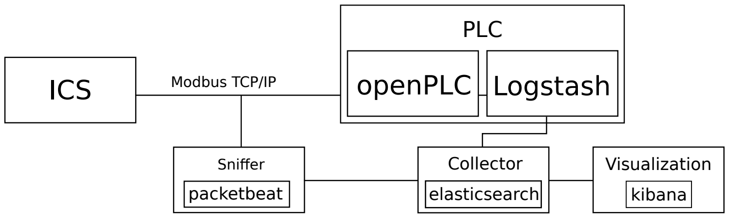


Fig. 2. Architectural Overview PLC Aggregation

The ICS communicates with the PLC we set up using openPLC [2]. The network traffic is gathered with packetbeat, the log inforamtion can be extracted with logstash. Elasticsearch serves as a collector, the findings can be visualized with kibana. These tools are provided by the company elastic [1].

5 Conclusion and Outlook

In this paper, we introduced a novel data aggregation and correlation model. We showed that it can be employed in multiple fields by presenting two different use cases. Furthermore, we spotlighted features of our model that meet the demands in IIoT. Those features can be used to help managing the rising amount of data of interconnected and intelligent systems. Several different tasks, such as IT-security or maintenance, motivate the necessity for sophisticated data management. Our model is able to support this increasinly complex task.

As a next step, we will develop connection operators to make correlation of events easier. We will enrich our list of data sources to get a more holistic overview of networks. Other than that, we plan to incorporate algorithms of machine learning into our framework to help the system find correlations between events and detect anomalies of any kind.

Acknowledgments This work has been supported by the Federal Ministry of Education and Research of the Federal Republic of Germany (Foerderkennzeichen KIS4ITS0001, IUNO). The authors alone are responsible for the content of the paper.

References

1. elastic. https://www.elastic.co/de/
2. OpenPLC. http://www.openplcproject.com/
3. OpenSecurityArchitecture. http://www.opensecurityarchitecture.org/cms/library/icon-library
4. Buchmann, A., Koldehofe, B.: Complex Event Processing. In: it - Information Technology. vol. 5. Oldenbourg Wissenschaftsverlag (2009)
5. Cugola, G., Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing. In: ACM Computer Surveys (2012)
6. Dodds, P., et al. (eds.): Sharable Content Object Reference Model (SCORM(TM)). Advanced Distributed Learning Initiative (2001)
7. Eckert, M., Bry, F.: Aktuelles Schlagwort: Complex Event Processing (CEP). Springer Verlag (2008)
8. Ficco, M.: Security Event Correlation Approach for Cloud Computing. In: International Journal of High Performance Computing and Networking (2013)
9. Gyllstrom, D., Diao, Y., Wu, E., Stahlberg, P., Chae, H.J., Anderson, G.: SASE: Complex Event Processing over Streams (2007)
10. Jiang, G., Cybenko, G.: Temporal and Spatial Distributed Event Correlation for Network Security. In: Proceedings of the American Control Conference (2004)
11. Kliger, S., Yemini, S., Yemini, Y., Ohsie, D., Stolfo, S.: A Coding Approach to Event Correlation (1993)
12. Luckham, D.: A Short History of Complex Event Processing (2007)
13. Luckham, D.C., Frasca, B.: Complex Event Processing in Distributed Systems (1998)
14. Robins, D.B.: Complex Event Processing (2010)
15. Wang, F., Liua, S., Liu, P., Bai, Y.: Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams (2005)
16. Wu, E., Diao, Y., Rizvi, S.: High-Performance Complex Event Processing over Streams (2006)

All links were last followed on January 3, 2016.

Elastic Allocation of Docker Containers in Cloud Environments

Matteo Nardelli

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

Abstract Docker containers wrap up a piece of software together with everything it needs for the execution and enable to easily run it on any machine. For their execution in the Cloud, we need to identify an elastic set of virtual machines that can accommodate those containers, while considering the diversity of their requirements. In this paper, we briefly describe our formulation of the Elastic provisioning of Virtual machines for Container Deployment (EVCD), which takes explicitly into account the heterogeneity of container requirements and virtual machine resources. Afterwards, we evaluate the EVCD formulation with the aim of demonstrating its flexibility in optimizing multiple QoS metrics.

Keywords: Container, Cloud computing, Resource allocation, QoS

1 Introduction

Docker[1] containers enable to package an application together with all of its dependencies and then run it smoothly in other environments. They exploit the operating system level virtualization, therefore multiple containers can co-exist and run in isolation on the same machine, thus improving resource utilization. Differently from virtual machines, containers are lightweight [4], because they bundle only the application dependencies while reusing the underlying operative system. For the execution, a container needs to be deployed on a hosting machine, which provides computing and memory resources. While doing this, we still want to exploit the Cloud computing principles, which promote the elastic usage of on-demand resources. This problem is named *container deployment problem* (or container allocation problem). If the deployment of a single container can be done easily, deploying lots of them, belonging to multiple applications with different requirements, can be complicated and might lead to resource shortage or resource under-utilization. In the literature only few solutions consider container features while determining their allocation (e.g., [3, 6, 9, 11]) and the most of them are characterized by different assumptions and optimization goals. Aside from the work presented in [6], there is no general formulation of the container deployment problem in the Cloud.

[1] `https://www.docker.com/`

In this paper, we investigate the problem of deploying containers in the Cloud. Specifically, we evaluate EVCD [8], a general formulation of the container deployment problem that determines the optimal allocation on virtual machines, acquired on-demand, while considering the QoS attributes of containers and virtual machines. Besides computing the initial deployment and the following runtime adaptation, EVCD provides a benchmark against which other deployment algorithms can be compared. With respect to our previous work [8], in this paper we show how EVCD can optimize different user-oriented QoS metrics, such as the deployment time and cost.

The remainder of this paper is organized as follows. We review related work in Sect. 2; in Sect. 3 we describe the system model and the problem under investigation, before formulating EVCD as an Integer Linear Programming problem in Sect. 4. We evaluate the flexibility of EVCD in Sect. 5 and conclude in Sect. 6.

2 Related Work

In literature a limited number of works provides a formal definition of the allocation problem for containers; however, due to NP-hardness of the problem, many heuristics have been proposed. As regards the modeling, in [1] the authors propose a constraint programming model that, differently from our approach, finds a feasible (but not optimal) deployment solution. The work most closely related to ours has been presented by Hoenisch et al. [6]. Their model accounts for container replication as well as their allocation, while considering several QoS attributes. We do not explicitly consider container replication, however we postpone to future work the extension of EVCD for considering it.

The existing heuristics aim at optimizing a diversity of utility functions, namely fairness, load balancing, network traffic, or energy consumption. The fairness of resource allocation is considered in [5,10]. In [5], Ghodsi et al. propose the Dominant Resource Fairness (DRF) policy, which works in a system containing different resource types (i.e., CPU, memory) and assigns them to containers in a Pareto-optimal manner. Then, Wang et al. [10] further generalize the notion of DRF to work with multiple heterogeneous servers. Considering a topology of communicating containers, Zhao et al. [11] propose a policy that minimizes the traffic exchanged using the network, while balancing the load among virtual machines. The minimization of energy consumption is considered in [3, 9]; these works propose a greedy placement scheme that allocates containers on the most energy efficient machines first. All these works focus on system-oriented metrics, whereas we consider user-oriented metrics, such as the deployment time and cost. However, EVCD provides a general framework for solving the deployment problem that can be easily extended to incorporate also those kind of metrics.

Proprietary solutions that support container allocation (e.g., Amazon ECS[2], Google Container Engine[3]) and open-source alternatives (e.g., Kubernetes[4],

[2] https://aws.amazon.com/ecs/
[3] https://cloud.google.com/
[4] http://kubernetes.io/

Docker Swarm[5]) usually bundle simple scheduling heuristics and also enable the execution of custom policies. Among the most common heuristics, we can find the round-robin policy, which tries to evenly use resources, and well-known heuristics that solve the bin packing problem, namely greedy best-fit and first-fit. Specifically, Docker Swarm, the official Docker component that allocates containers on a centralized pool of resources, includes a bin packing policy and a strategy that balances the number of containers among computing nodes. In our previous work [8], we used EVCD to compare some of these heuristics (i.e., round-robin, greedy first-fit) in terms of achievable QoS performance.

3 System Model and Problem Statement

Devising an optimal container deployment strongly depends on the assumptions made about the domain it will be applied to. In this section, we provide a formal description of the domain entities: containers and virtual machines.

Software Container Model. A Docker container is an instance of an *image*, which represents a container snapshot and contains all the data needed for its execution. For efficiency reasons, an image is structured as a series of layers (e.g., libraries, custom files), where each one can be downloaded independently from the others. We define the set of containers as S. A container $s \in S$ is characterized by the following QoS attributes: C_s, the number of required CPUs; D_s^{sc}, the startup time; M_s, the amount of required memory; and I_s, the set of image layers needed for its instantiation. We assume $I_s \subseteq I$, where I is the set of all the available containers images. Each image layer $i \in I$ is characterized by a weighting factor l_i which represents the size of data composing the layer.

Virtual Machine Model. A virtual machine (VM) hosts and executes containers with respect to its capabilities. Being a Cloud resource, a VM can be acquired and released as needed and paid for the amount of, albeit partially, consumed Billing Time Units (BTU). Finally, although in theory unlimited, we assume the number of leasable virtual machines to be limited in a certain time period. Let V be the set of all VMs, including the active (leased) ones and the leasable ones. A virtual machine $v \in V$ has the following QoS attributes: C_v, the amount of available CPUs; DR_v, the download data rate of v; M_v, the available memory capacity; P_v, the cost per BTU; and I_v, with $I_v \subseteq I$, the set of image layers available in v without downloading them from an external repository.

Container Deployment Problem. To solve the deployment problem, we need to determine a mapping between the set of containers S and the set of virtual machines V in a such a way that all constraints are fulfilled. We investigate the initial deployment as well as its adaptation at runtime, therefore we solve EVCD periodically, every τ unit of time. We model the container deployment with binary variables $x_{s,v}$, $s \in S$, $v \in V$: $x_{s,v} = 1$ if s is deployed on v and $x_{s,v} = 0$ otherwise. The variables z_v denote whether $v \in V$ is active and hosts at least one container. Relying on the deployment configuration determined at

[5] https://www.docker.com/products/docker-swarm

time $t - \tau$, we also define the following auxiliary binary variables: a_v, which indicates whether $v \in V$, turned off in $t-\tau$, has to be activated in t; $a_{s,v}$, which indicates whether s is deployed on a newly activated virtual machine v (i.e., with $a_v = 1$); and $\delta_{s,v}$, which indicates whether, in $t - \tau$, $s \in S$ was not allocated or was allocated on $u \neq v$, with $u \in V$.

4 Elastic Provisioning Model

In this section we present our formulation of the EVCD problem as an Integer Linear Programming (ILP) model. Due to space limitations, we do not report the full formulation of EVCD, which can be found in [8]. EVCD is solved periodically, every τ unit of time. We model as QoS metrics the deployment time $D(\boldsymbol{x}, \boldsymbol{z})$ and cost $C(\boldsymbol{x}, \boldsymbol{z})$. The former represents the time needed to deploy every container in S, whereas the latter represents the monetary cost of the virtual machines leased for executing the containers. Since the relative importance of these metrics depends on the utilization scenario, EVCD provides a general formulation that can be aimed at optimizing specific QoS attributes. Therefore, the objective function of EVCD is the minimization of $F(\boldsymbol{x}, \boldsymbol{z})$, which is defined as a weighted sum of the normalized QoS attributes:

$$F(\boldsymbol{x}, \boldsymbol{z}) = w_d \frac{D(\boldsymbol{x}, \boldsymbol{z}) - D_{\min}}{D_{\max} - D_{\min}} + w_c \frac{C(\boldsymbol{z}) - C_{\min}}{C_{\max} - C_{\min}} \tag{1}$$

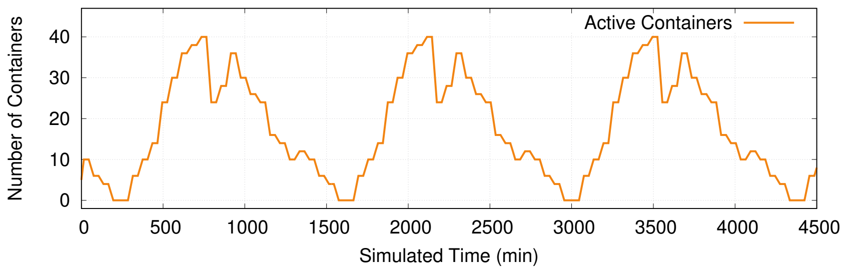
where w_d, w_c, with $w_d,\ w_c \geq 0$, $w_d + w_c = 1$, are weights for the different QoS attributes, and $D_{\max}$ ($D_{\min}$) and $C_{\max}$ ($C_{\min}$) denote, respectively, the maximum (minimum) value for the overall expected deployment time and cost. Observe that these normalization factors can be computed by solving other optimization problems or can be approximated relying on previous experiments or on statistical analysis of the runtime execution. The overall deployment time $D(\boldsymbol{x}, \boldsymbol{z})$ accounts for the time needed to spawn new VMs, retrieve container images, and finally start the containers. It can be formally defined as:

$$D(\boldsymbol{x}, \boldsymbol{z}) = \sum_{s \in S} \sum_{v \in V} \left(D_v^{sv} a_{s,v} + \sum_{i \in I_s \setminus I_v} \frac{l_i}{DR_v} x_{s,v} + D_s^{sc} \delta_{s,v} \right) \tag{2}$$

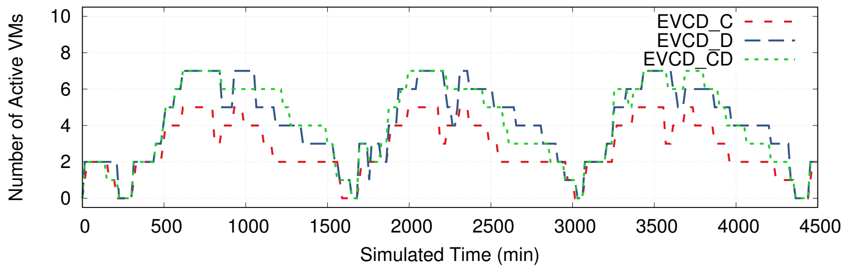
where D_v^{sv} is the startup time of a new VM, considered only if a new one is needed, $\sum_i \frac{l_i}{DR_v}$ represents the time needed to download the images I_s not yet on v, and D_s^{sc} is the startup time of s, if s was not already running on v. The cost $C(\boldsymbol{x}, \boldsymbol{z})$ includes the leasing of the new VMs and the renewing the expired ones which are still needed. Relying on the activation vector $\boldsymbol{z}$ and on the auxiliary variables a_v, we have that:

$$C(\boldsymbol{z}) = \sum_{v \in V} P_v a_v + \sum_{v \in V^{exp}} P_v z_v \tag{3}$$

where the first term on the right end side accounts for the cost of newly acquired VMs, and the second one accounts for renewing the expired leasings, if needed



(a) Number of active containers



(b) Number of active virtual machines

Figure 1: Elastic provisioning of virtual machines when EVCD receives a varying demand for resource allocation by containers.

(i.e., if the related VM is still used). The set $V^{exp} \subseteq V$ includes the VMs whose leasing is going to expire between the current time t and the next execution of EVCD in $t + \tau$.

The full optimization in [8] includes constraints that limit the number of containers deployable on a VM with respect to the available resources as well as the formal definition of the auxiliary variables.

5 Experimental Results

To evaluate the EVCD model, we simulate its execution in a system that receives containers and allocates them on VMs. The aim of this experiment is to show the flexibility of EVCD, which can elastically acquire and release VMs to host and execute software containers while optimizing the deployment time of containers, the cost of leased VMs, or a combination thereof.

We solve EVCD with CPLEX©[6], the state-of-the-art solver for ILP problems, on a machine with 4 CPUs and 16 GB RAM. We simulate the execution of EVCD every $\tau = 15$ minutes. Each container requires unitary resources (i.e., CPU,

[6] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Table 1: QoS metrics and VMs acquired under different configurations of EVCD

Average values of the QoS metrics of interest

QoS metric	**EVCD_C**	**EVCD_D**	**EVCD_CD**
Cost of leased VMs per hour	3.8	4.9	4.1
Deployment time per container	24.0 s	13.6 s	14.8 s

Relative number of used virtual machines

VM type	**EVCD_C**	**EVCD_D**	**EVCD_CD**
type A	23.4%	44.8%	36.8%
type B	76.6%	55.2%	63.2%

memory) and has a lifespan of $L = 30$ minutes. A container depends on a set of image layers, whose cardinality is uniformly chosen between 2 and 4; this set includes 70% of existing images (if any) and 30% of new images. Each image layer has a size l_i uniformly defined in $[200, 800]$ MB. The startup time D_s^{sc} of $s \in S$ ranges uniformly in $[8.5, 11.5]$ s. Two type of VMs are available in V: *type A* with 4 CPUs, 16 GB of RAM, and $P_v = 1$; and *type B* with 8 CPUs, 32 GB of RAM, and $P_v = 1.7$. Similarly to the most popular commercial solutions, the BTU is 60 minutes. The startup time D_v^{sv} of $v \in V$ ranges uniformly in $[85, 115]$ s, in accordance with [7]. During the experiment, the number of containers requiring resources fluctuates between 0 and 20 during the timespan of a (simulated) day, and this pattern is repeated for the following two days. Figure 1a shows the number of active containers during the whole experiment; being $L \geq \tau$, the number of active container can be greater than 20.

We evaluate the effects on the containers deployment of three different configurations of EVCD, namely EVCD_C, EVCD_D, and EVCD_CD. **EVCD_C** deploys containers by minimizing, as QoS metric, the cost $C(\boldsymbol{x}, \boldsymbol{z})$, i.e., it solves EVCD with F parametrized with weights $w_c = 1$ and $w_d = 0$. **EVCD_D** minimizes the deployment time $D(\boldsymbol{x}, \boldsymbol{z})$, by solving EVCD with $w_d = 1$ and $w_c = 0$. Finally, **EVCD_CD** minimizes both the QoS metrics, by solving EVCD with $w_d = w_c = 0.5$, and normalization terms $D_{\min} = 8.5$ s, $D_{\max} = 152.1$ s, $C_{\min} = 0$, and $C_{\max} = 10.1$. These values result from preliminary experiments. Figure 1b shows the number of active VMs during the experiment, whereas Table 1 reports the average values of the considered QoS metrics and the relative number of used VMs per type. From Fig. 1b we can see that, although every configuration of EVCD acquires and releases VMs to satisfy the incoming load, each of them follows a different strategy. Differently from the other configurations, EVCD_C tries to use as few VMs as possible and, if needed, prefers *type B* VMs (76.6% of the time, see Table 1), because of their economy of scale. However, this leads to the highest deployment time, which is higher then the optimal one of about 76% (see Table 1). EVCD_D neglects the differences in terms of price between VMs of *type A* and *type B*, which are used almost equally. This strategy has the highest cost (29% higher than the optimal one), however it obtains the lowest possible deployment time, because it allocates containers on VMs that already

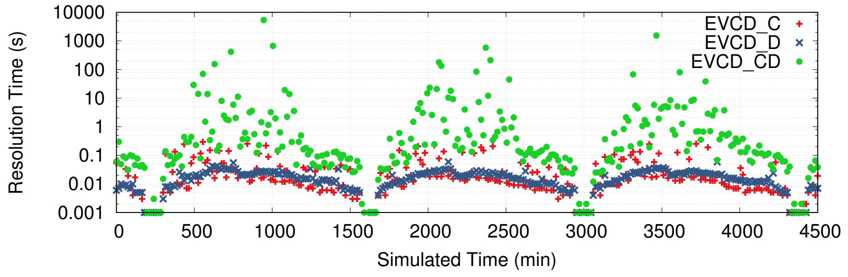


Figure 2: Resolution time of EVCD.

host some of the needed image layers, thus reducing the container startup time. EVCD_CD finds a trade-off between the previous strategies, as can be seen from Table 1. It optimizes both the QoS metrics, achieving a deployment time and cost about 9% and 8% higher than the optimal values, respectively. Moreover, this strategy shows that, by selecting the weights of F according to the user preferences, EVCD can be aimed at optimizing different QoS metrics of interest. Observe that any other combination of weights w_c, w_d lead to configurations that lie in between EVCD_C and EVCD_D. Determining the best trade-off between deployment time and cost depends on the relative importance of these QoS metrics on the utilization scenario (e.g., different user preferences).

On EVCD Resolution Time. We now discuss about the *resolution time* of EVCD and its relationship with the optimization goals. We consider as resolution time the time needed to compute the exact solution of the ILP problem. During this experiment, the maximum number of managed virtual machines (both active and leasable) in V is 19, whereas the maximum number of active containers in S is 40. Figure 2 reports the resolution time of EVCD for the different configurations (i.e., EVCD_C, EVCD_D, and EVCD_CD) during the experiment. In general, we observe that the resolution time is influenced by the size of the problem as well as by the optimization function F resulting from the different set of weights. Optimizing a single QoS attribute, i.e., solving EVCD_C or EVCD_D, is less computationally demanding and has better performances; as can be seen from Fig. 2, the resolution time of EVCD_C and EVCD_D is always below 1 s. Conversely, solving a multi-objective optimization problem is harder and produces higher resolution times with greater fluctuations with respect to the previous configurations. The complexity of CPLEX does not easily reveal the motivations behind the number and amplitude of these fluctuations. During the whole experiment, the 95th percentile of the resolution time for EVCD_C and EVCD_D is 189 ms and 33 ms, respectively, whereas EVCD_CD has a 95th percentile of about 24.6 s, i.e., two order of magnitude higher.

It can be demonstrated that the deployment problem is NP-hard (see [2]), therefore EVCD does not scale well as the problem instance increases in size. Nevertheless, by determining the optimal deployment of containers over an elastic set of VMs and evaluating the subsequent runtime reconfigurations, EVCD provides a benchmark for evaluating heuristics, for developing new ones, and for identifying the most suitable ones with respect to specific optimization objectives.

6 Conclusion

In this paper we have described and evaluated EVCD, a formulation of the elastic provisioning of virtual machines for container deployment. EVCD is a general and flexible model that can be conveniently configured to optimize different QoS metrics. Aside computing the initial allocation, EVCD can adapt the containers deployment at runtime, if needed. The experimental evaluation has shown the flexibility of the proposed model, which has optimized the deployment time of containers, the monetary cost for their execution, and a combination thereof. As future work, we plan to extend the formulation of EVCD to include other QoS attributes (e.g., network traffic, resource utilization) and the runtime replication of containers. Moreover, we plan to develop efficient heuristics to deal with large instances of the container deployment problem for the initial deployment and to support runtime reconfigurations.

References

1. Abdelbaky, M., Diaz-Montes, J., Parashar, M., et al.: Docker containers across multiple clouds and data centers. In: Proc. of IEEE/ACM UCC 2015 (2015)
2. Cardellini, V., Grassi, V., Lo Presti, F., Nardelli, M.: Optimal operator placement for distributed stream processing applications. In: Proc. of ACM DEBS '16 (2016)
3. Dong, Z., Zhuang, W., Rojas-Cessa, R.: Energy-aware scheduling schemes for cloud data centers on google trace data. In: Proc. of IEEE OnlineGreenComm 2014 (2014)
4. Felter, W., Ferreira, A., Rajamony, R., et al.: An updated performance comparison of virtual machines and linux containers. In: Proc. of IEEE ISPASS 2015 (2015)
5. Ghodsi, A., Zaharia, M., Hindman, B., et al.: Dominant resource fairness: Fair allocation of multiple resource types. In: NSDI. vol. 11 (2011)
6. Hoenisch, P., Weber, I., Schulte, S., et al.: Four-fold auto-scaling on a contemporary deployment platform using docker containers. In: Proc. of ICSOC 2015 (2015)
7. Mao, M., Humphrey, M.: A performance study on the vm startup time in the cloud. In: Proc. of IEEE CLOUD 2012 (2012)
8. Nardelli, M.: Elastic provisioning of virtual machines for container deployment. In: Submitted to ACM/SPEC ICPE '17 (2017)
9. Piraghaj, S.F., Dastjerdi, A.V., Calheiros, R.N., et al.: A framework and algorithm for energy efficient container consolidation in cloud data centers. In: Proc. of IEEE DSDIS 2015 (2015)
10. Wang, W., Li, B., Liang, B.: Dominant resource fairness in cloud computing systems with heterogeneous servers. In: Proc. of IEEE INFOCOM 2014 (2014)
11. Zhao, Z., Mandagere, N., Alatorre, G., et al.: Toward locality-aware scheduling for containerized cloud services. In: Proc. of IEEE Big Data 2015 (2015)

Choreographies are Key for Distributed Cloud Application Provisioning

Oliver Kopp[1] and Uwe Breitenbücher[2]

[1] IPVS, Universität Stuttgart, Germany
kopp@ipvs.uni-stuttgart.de
[2] IAAS, Universität Stuttgart, Germany
breitenbuecher@iaas.uni-stuttgart.de

Abstract. The automation of Cloud application provisioning is one of the most important key success factor for Cloud Computing. In case a complex composite Cloud application has to be provisioned across multiple different private Clouds, a single centralized provisioning engine or workflow is not possible: For security reasons, private Clouds typically do not expose their internal provisioning APIs to the outside to make them callable by an external service. Thus, distribution of provisioning logic across multiple is required. This paper envisions that choreographies can be used to distribute the provisioning logic.

Keywords: choreography, provisioning, cloud computing

1 Overview

Cloud computing gains more and more attention due to its economical, organizational, and technical benefits. The reason for this evolution lies in the essence of Cloud computing—the industrialization of IT [1]: application provisioning, operation, management, and termination are automated as much as possible and follow well-defined processes to provide industrial properties for virtualized infrastructure, middleware, and software as a service. To enable this, Cloud providers employ Cloud management software to virtualize, manage, and operate their physical infrastructure. One part of this management is responsible for rapid on-demand provisioning of parts of a multi-Cloud application for customers. It has been shown that not all management tasks can be done in a declarative way [2]. Moreover, many applications need to consume different services for their particularities that are not completely provided by one single Cloud provider [6]. Thus, these applications need to be distributed across several different Clouds to gain the needed functionality. Distributing components of one application onto multiple Clouds, however, increases the complexity of provisioning: Provisioning applications on a single Cloud (based on internal provisioning engines) helps the Cloud provider to secure the system: most of the required management operations can be executed locally by provisioning engines behind strict firewalls and only coarse-grained fascade APIs need to be exposed to the outside. This changes if the application is distributed across multiple Clouds: the provisioning on different Clouds needs to be orchestrated by an external provisioning engine. Thus, Cloud providers have to expose also the low-level management operations, interfaces, and APIs to make them accessible from the outisde. Figure 1 shows this kind of provisioning for an application hosted on a Tomcat as WAR file. First, a virtual machine has to be instantiated, then a Tomcat has to be installed. Afterwards, the WAR file has to be deployed and configured. The calls to the respective management operations have to pass the firewall of the local Cloud provider, which might not be allowed.

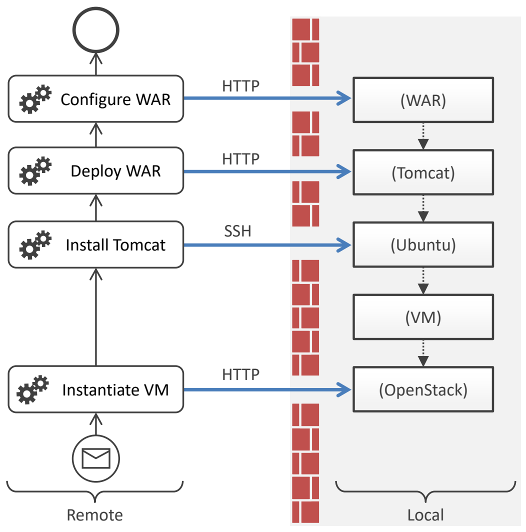


Fig. 1: Orchestration-based Provisioning

2 The Vision

We have the vision of modelling the provisioning of distributed multi-Cloud applications as a choreography model, wherein each participant flow executes the provisioning of one part of the application on one provider.

As a consequence, we need a means to coordinate the provisioning of multi-Cloud applications in a way that the execution of provisioning logic remains on the side of Cloud providers in order to avoid the low-level management functionalities must be made accessible from the outside. Our proposed solution is to use choreographies [5] for application provisioning: the provisiong logic has to be split among the participants, each representing a local Cloud provider. Figure 2 presents such a model for the example. Now, the remote workflow sends

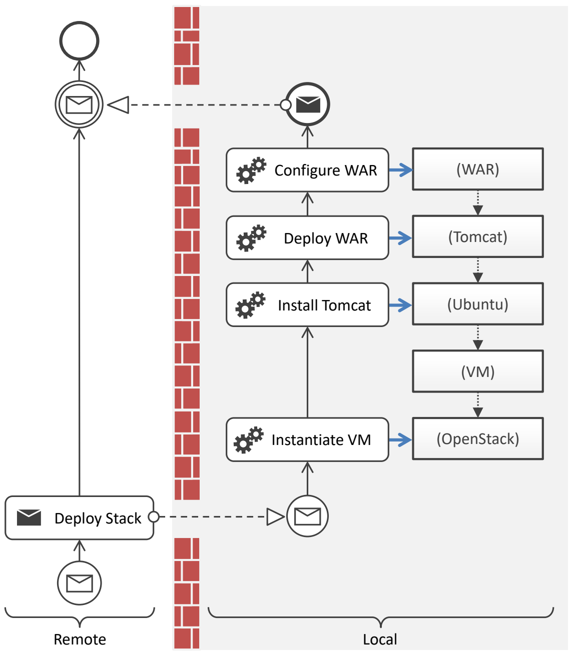


Fig. 2: Choreography-based Provisioning

a single message triggering the provisioning within the local Cloud provider. The provisioning is implemented by a workflow running within the local Cloud provider.

3 Discussion

Herry et al. [3] generate provisioning workflows out of given constrains. Our approach relieas on human-generated choreography models. The infrastructure for executing the provisioning workflows does not need to be available all the time: Vukojevic-Haupt et al. [7] showed hat it is also possible to provision that middleware on demand.

Another approach is to model a single provisioning workflow and to split this workflow according to the distribution of the topology. For the split, new coordinators might be required [4]. When using a choreography model, the provisioning workflows and their coordination can be generated easily as the coordination of participants is inherently specified by the choreography.

The idea presented in the paper is not tied to concrete languages or tools. To validate the approach, we plan to (i) enable swimlanes in our current modeling tool and (ii) to show that the approach is feasable in our OpenTOSCA eco system.

Acknowledgments This work is funded by the BMWi project SmartOrchestra (01MD16001F).

References

1. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (May 2012)
2. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In: International Conference on Cloud Engineering (IC2E 2014). pp. 87–96. IEEE (Mar 2014)
3. Herry, H., Anderson, P., Rovatsos, M.: Choreographing Configuration Changes. In: Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013). pp. 156–160. IEEE (Oct 2013)
4. Khalaf, R., Leymann, F.: Coordination for fragmented loops and scopes in a distributed business process. Information Systems 37(6), 593–610 (2012), bPM 2010
5. Peltz, C.: Web Services Orchestration and Choreography. IEEE Computer 36(10), 46–52 (2003)
6. Petcu, D.: Multi-cloud: expectations and current approaches. In: MICAS Workshop. pp. 1–6. MultiCloud '13, ACM, New York, NY, USA (2013)
7. Vukojevic-Haupt, K., Haupt, F., Leymann, F., Reinfurt, L.: Bootstrapping complex workflow middleware systems into the cloud. In: 2015 IEEE 11th International Conference on e-Science. Institute of Electrical and Electronics Engineers (IEEE) (Aug 2015)

Benchmark Proposal for Multi-Tenancy in the Database Layer

Philipp Neugebauer[1], Christian Maier[2] and Alexander Bumann[3]

[1] innoQ Deutschland GmbH, Kreuzstraße 16, D-80331 München
philipp.neugebauer@innoq.com
[2] Information Systems and Services, University of Bamberg
christian.maier@uni-bamberg.de
[3] nubibase GmbH, Silbersteinstraße 14, D-97424 Schweinfurt
a.bumann@nubibase.de

Abstract. The cloud is often utilized with the hope to increase the IT budget efficiency. Software as a Service in combination with its key feature multi-tenancy shines here brightly but its adoption is complexed by the multiplicity of solutions. In detail, multiple multi-tenancy approaches meet many possible database systems requiring a reliable comparison to find the optimal match. This article briefly explains multi-tenancy, its benefits and implementations and indicates through the results of an extensive literature review a missing benchmark of multi-tenancy approaches. It proposes and describes the benchmark and its setup to gain reliable results of the space usage and performance.

Keywords: cloud, SaaS, multi-tenancy, database, benchmark

1 Introduction

An increasing number of companies try to address their business challenges and opportunities with their IT budget more efficiently in response to growing IT budgets or IT budget constraints.[1,2] Restrictively, big business solutions and services do not match requirements and also budgets for small and middle-sized companies leaving a business gap and opportunity [1]. A possible solution to their problem is provided by the cloud, especially Software as a Service (SaaS) in combination with multi-tenancy. The application of multi-tenancy enables the better utilization of hardware and economies of scale for IT service providers [2, 3]. It also allows the pooling of more tenants on a single server, resulting in either a smaller price or a bigger gain and therefore a better competitiveness [2]. Small or middle-sized companies on the other hand are then able to optimize or even reduce their IT budget through the adoption of these services. Big companies can take advantage of multi-tenancy by merging their internal services into a single database or application to reduce their maintenance staff and costs [1].

[1] http://diginomica.com/2016/12/21/it-spending-and-staffing-in-2017-computer-economics-on-the-cios-cloud-predicament/
[2] http://www.zdnet.com/article/research-it-budgets-rising-for-many-in-2015/

Multi-tenancy is the primary feature of SaaS and enables the handling of multiple independent tenants in a single instance on a single hardware and software infrastructure while allowing individual configuration to the tenant's needs [3, 4, 5]. It captivates by the combination of efficiency improvement and the enabling of customization which provides a broad range of application targets. The most efficient application of multi-tenancy can improve the served number of tenants from about a dozen to more than hundred per machine and therefore enable huge cost savings [6, 7].

Companies have different possibilities to utilize multi-tenancy and thereby to improve their IT budget efficiency. Various approaches have been proposed in a lot of articles [1, 5, 7, 8, 9] for the adoption of multi-tenancy in the database layer proving an academic interest in the research field. Thus, diverse implementations for different database solutions are available, but even though some of them have been tested, contractionary results have been released which exacerbates the solution selection [6, 8, 9]. Unfortunately, neither a guideline for the selection of the best matching multi-tenancy approach nor a summary of all implementations exists to simplify the identification of possible solutions. The development of such a guideline requires a reliable benchmark with criterias like space usage and performance to test and compare the multi-tenancy approaches to enable the selection of the best matching solution for the scenario.

This article proposes the setup for the benchmark so that the guideline for the selection of multi-tenancy approaches can be created. It covers first the different multi-tenancy implementations and reasons why the existing work cannot be used for the guideline development. In the following, the ideas for the settings and configuration of the benchmark for multi-tenancy approaches are explained. Lastly, the article is concluded and future work areas and plans are presented.

2 Multi-Tenancy Background and Related Work

This section briefly explains the different levels of multi-tenancy in the database layer. It approaches their implementation opportunities and lastly justifies the required creation of a new benchmark.

Separate databases provide each tenant its own database instance that is adaptable to their needs but multiple tenants share the same machine [6, 10, 11]. The *shared database* provides each tenant its own tables, but all share the same database process [7, 11]. Lastly, *shared table* approaches [7, 11] preserve data from many tenants in the same tables and each row is marked with a tenant id [11]. *Shared tables* are not a single approach, they represent a category of various implementations most efficiently addressing the problem of multi-tenancy adoption. They avoid data replication, support client customizations and encourage business changes while serving the highest number of tenants through their lowest overhead [6, 10]. On the downside, the solutions often turn the database into a dumb data repository and many features like query optimization or indexes must be reimplemented [11, 12]. Additionally, the security must be handled outside the database and generic columns can only used if sparse tables are handled efficiently [11].

In summary, 22 *shared table* approaches were identified in the extensive literature review. The standardized benchmarking of *shared table* approaches is complicated by the required adaptions of the initial benchmark schema to each approach because they utilize custom schemas. Additionally, the merge of the database tables of all tenants requires an extendable approach schema and therefore the extensibility and the efficiency of its implementation must be determined. The different approach schemas may also vary about the efficiency of their space usage so that the space usage must be analyzed specifically for *shared table* multi-tenancy approaches. Lastly, it must be checked if the approaches can be implemented in the specific database system or must be excluded for some or all systems. The relational database systems to be benchmarked must be added to the complexity of the missing overview, resulting in a non trivial n to n problem which is illustrated in Table 1.

Database	Basic Tables	Private Tables	Extension Tables	XML Columns	Index Tables	Clustering Extension Tables	Elastic Extension Tables	Preallocated Fields	Universal Table	Multiple Sparse Tables	Native Support of Multi-Tenancy	M-store	Pivot Tables	Chunk Tables	Chunk Folding	Adaptive Schema Design Method	Cluster Based Schema Design	Interpreted Record	Multi-tenant Shared Tables	FLEXSCHEME	Flexible Relational Data Model
⋮																					

Table 1: Problem Visualization

All identified and implementable approaches need to be comprehensibly and repeatably compared by their space usage and performance to gain the information for the determination of the optimal implementation. Such a test environment is realized in a benchmark which are created in the articles of Krebs et al. [3] and Kiefer et al. [13] but they lack the full support for *shared table* approaches. Krebs et al. [3] developed an extension for the in the meantime as obsolete marked TPC-W[3] and is only able to test the very simple *basic table* approach. The MulTe framework designed by Kiefer et al. [13] can only examine *separate databases* and *shared databases*. Benchmark standards of the Transaction Processing Performance Council (TPC)[4] cannot be applied to *shared table* multi-tenancy approaches since they support only single tenant databases and therefore provide no tenant configurable schema. Hui et al. [2] based their test settings on the TPC-C and TPC-H frameworks but just briefly described them and also

[3] `http://www.tpc.org/tpcw/`
[4] `http://www.tpc.org/`

optimized them to their test scenario. However, some of their test specifications inspired this proposed benchmark and were adopted. In conclusion, no existing benchmark can be utilized for the testing of *shared table* multi-tenancy scenarios.

3 Benchmark Proposal

This section covers the proposed benchmark. The complete benchmark must be automatically executable and easily extendable to cover new *shared table* approaches to finally develop guidelines for the simple selection and implementation of the best matching approach. In the beginning, only the *shared table* approaches are addressed due to their best cost efficiency. First, the benchmark background is approached by the influences to this proposal and the description of TPC. Next, the general setup of the benchmark is explained. Third, the test environment is specified in more detail. The settings for result measurement conclude the section.

Existing research of Hui et al. [2] and Aulbach et al. [8] inspired the setup of this benchmark. Hui et al. [2] contributed the foundation of TPC-H, the concurrent access query number and the measurement of space usage and performance, while Aulbach et al. [8] partially utilized the query statement numbers and also the concurrency aspect. TPC-H was selected as benchmark base because it aims on concurrent data manipulation and data of large volumes and broad industry-wide relevance which are the characteristics of cloud scenarios. Moreover, the service for many customers leads to big data volumes and requires the handling of concurrent accesses. Since TPC-H approaches normal business requirements, has an industry-wide relevance and is easily implementable in relational databases, it also provides a fair benchmark base. The selection is further justified by Kiefer et al. [13] who also based their benchmark on TPC-H. The Transaction Processing Performance Council is a community which provides a wide range of database benchmarks to enable effective performance comparisons between them. TPC frameworks are currently not applicable to multi-tenancy approaches as already stated, so that only general guidelines of TPC-H could be adopted and were then applied to the *shared table* context: In the beginning, the database tables are created. Afterwards, the data is generated and lastly the test queries are executed.

Mapping these guidelines to the multi-tenancy context results in the initial setup of all tenants' general schema. The test data is generated and populated into the database to ensure the same base data for all tested approaches to avoid any result biasing. In the next step, the specific approach tables regarding the tenants' requirements are created and the base data is mapped into them. In the last step, the concurrent benchmark queries are executed and benchmarked.

The schema is specifically chosen for the testing of multi-tenancy and comprises only the three tables orders, lineitems and parts to simulate multi-tenant characteristics. The schema of TPC-H is therefore simplified and to multi-tenancy adapted to focus on the determination of the performance in terms of response time allowing for the extensibility of the approaches. The targeted scenario is a service provider who serves multiple small clients by hosting and managing their data and aims for the long-tail of business applications. It does not change the stored data volume of the big single tenant application of the TPC-H but splits them between between many tenants. While only the shared attributes of all tenant groups are displayed in Figure 1, the schemas of the other groups are enriched for their use cases. The benchmark scenario will be configurable between 100, 250 and 500 tenants, each having ordered 3000 times with four lineitems. The catalog contains 187500 parts so that the benchmark with 500 tenants will run against more than 7.5 million records in total. The tenant number is based on the assumption that the usage of *shared tables* must provide a significant increase of served tenants to warrant the effort for its adoption. The other ratios are taken from the TPC-H benchmark whose selection is justified in section 2. The lineitem table is used to regard the different tenant requirements and has a customized column length for each tenant group. Five tenant groups are introduced in the tests meeting different business requirements which results in varying table column numbers. Therefore, also different ratios of these groups will be considered to represent multiple business scenarios. The first groups uses only a very small subset of the possible attributes, the second employs just slightly less attributes than the normal group which utilizes the general schema. The fourth group lacks a bit more attributes than the normal group and the fifth group utilizes much more attributes than the all other groups to satisfy their business requirements.

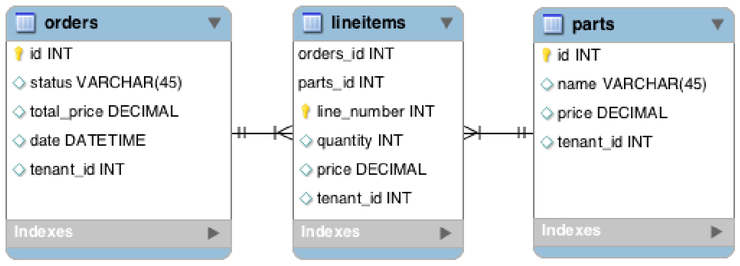


Fig. 1: Evaluation Schema

The measurement and resulting metrics are conducted with the execution of SELECT, INSERT, UPDATE and DELETE queries. SELECT queries must be more often executed to provide helpful results leading to 100 and 1000 statements, while INSERT, UPDATE, DELETE queries allow already good comparisons at 50 and 100 statements. The benchmark queries are inspired by the queries of TPC-H but primary focus on the performance testing of the characteristics of *shared tables* or multi-tenancy instead of specific business logic. In consequence, queries must be executed on both the attributes shared by all tenant groups and custom extension attributes since they are a key feature for the enabling of multi-tenancy and their impact needs to be analyzed. A practically relevant SaaS scenario requires additionally the concurrent execution of tenants' queries which is met through queries of a configurable number of tenants considering the tenant groups' ratio. The minimum concurrent load will be 10% of the tenants and can be increased up to 50%. Lastly, same and different table queries must be executed to reveal the load balancing capability of the multi-tenancy approaches. As result, the strength and weaknesses of the approaches regarding the storage usage as well as the response time of reads and writes can be demonstrated and analyzed.

4 Conclusion

This article indicates through the results of an extensive literature review a missing benchmark of multi-tenancy approaches. It also proposed the setup for a reliable and useful multi-tenancy benchmark for *shared table* approaches to close the identified information gap. The benchmark is inspired by existing work and seized their ideas for a general *shared table* benchmarking scenario to ensure meaningful results for all identified approaches in relevant scenarios. Future research does not need to collect basic information about multi-tenancy anymore and can either extend the proposed benchmark or start its development. This will finally enable the creation of an guideline and so simpler adoption decisions for companies as well as avoid costly wrong decisions. The development focuses on relational database systems like MySQL and PostgreSQL, but the support for NoSQL databases is considered as a possible and useful option because the latter enables the utilization of new technologies which cannot be realized in relational databases. The benchmark will be theoretically tested against the findings of Sim et al. [14] and Huppler [15]. Beside the inclusion of *separate databases* and *shared databases*, the benchmark could also consider the handling of metadata and middleware in future to take these effects into account.

References

[1] O. Schiller, B. Schiller, A. Brodt, and B. Mitschang. “Native Support of Multi-tenancy in RDBMS for Software as a Service”. In: *Proceedings of the 14th International Conference on Extending Database Technology*. EDBT/ICDT ’11. Uppsala, Sweden: ACM, 2011, pp. 117–128.

[2] M. Hui, D. Jiang, G. Li, and Y. Zhou. “Supporting Database Applications as a Service”. In: *Data Engineering, 2009. ICDE ’09. IEEE 25th International Conference on.* 2009, pp. 832–843.

[3] R. Krebs, A. Wert, and S. Kounev. “Multi-tenancy Performance Benchmark for Web Application Platforms”. In: *Proceedings of the 13th International Conference on Web Engineering.* ICWE’13. Aalborg, Denmark: Springer-Verlag, 2013, pp. 424–438.

[4] T. Kwok, T. Nguyen, and L. Lam. “A Software As a Service with Multi-tenancy Support for an Electronic Contract Management Application”. In: *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2.* SCC ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 179–186.

[5] H. Yaish, M. Goyal, and G. Feuerlicht. “An Elastic Multi-tenant Database Schema for Software as a Service”. In: *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, DASC 2011, 12-14 December 2011, Sydney, Australia.* IEEE, 2011, pp. 737–743.

[6] L. Heng, Y. Dan, and Z. Xiaohong. “Survey on Multi-Tenant Data Architecture for SaaS”. In: *International Journal of Computer Science Issues.* Vol. 9. 2012.

[7] W. Lehner and K.-U. Sattler. “Virtualization for Data Management Services”. English. In: *Web-Scale Data Management for the Cloud.* Springer New York, 2013, pp. 13–58.

[8] S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold. “A Comparison of Flexible Schemas for Software As a Service”. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data.* SIGMOD ’09. Providence, Rhode Island, USA: ACM, 2009, pp. 881–888.

[9] S. Pippal and D. Kushwaha. “A Simple, Adaptable and Efficient Heterogeneous Multi-tenant Database Architecture for ad hoc Cloud”. English. In: *Journal of Cloud Computing* 2.1, 5 (2013).

[10] F. Chong, G. Carraro, and R. Wolter. *Multi-Tenant Data Architecture.* MSDN Library, Microsoft Corporation. 2006.

[11] D. Jacobs and S. Aulbach. “Ruminations on Multi-Tenant Databases”. In: *BTW Proceedings, volume 103 of LNI.* GI, 2007, pp. 514–521.

[12] S. Aulbach, M. Seibold, D. Jacobs, and A. Kemper. “Extensibility and Data Sharing in Evolving Multi-tenant Databases”. In: *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany.* Ed. by S. Abiteboul, K. Böhm, C. Koch, and K. Tan. IEEE Computer Society, 2011, pp. 99–110.

[13] T. Kiefer, B. Schlegel, and W. Lehner. “MulTe: A Multi-Tenancy Database Benchmark Framework”. In: *Selected Topics in Performance Evaluation and Benchmarking.* Vol. 7755. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 92–107.

[14] S. E. Sim, S. Easterbrook, and R. C. Holt. “Using Benchmarking to Advance Research: A Challenge to Software Engineering”. In: *Proceedings of the 25th International Conference on Software Engineering.* IEEE Computer Society. 2003, pp. 74–83.

[15] K. Huppler. “The Art of Building a Good Benchmark”. In: *Performance Evaluation and Benchmarking.* Springer, 2009, pp. 18–30.

Modeling and Analysis of Sustainability in Product Life Cycles

Andreas Fritsch

Karlsruhe Institute of Technology (KIT)
Institute of Applied Informatics and Formal Description Methods (AIFB)
andreas.fritsch@kit.edu

Abstract. This paper describes a work-in-progress, design-oriented approach to support the analysis of sustainability in product life cycles. The envisioned system provides a modeling method for structuring and collecting information about a product life cycle. By adapting analysis methods from the research field of environmental and social life cycle assessment, it further allows to explore and identify sustainability risks.

Keywords: Domain-Specific Modeling Languages, Enterprise Modeling, Sustainability, Life Cycle Assessment

1 Introduction

The recently adopted United Nations resolution ”Transforming our world: the 2030 Agenda for Sustainable Development” [12] is an example of the growing global awareness for environmental, economic and social problems. The complexity of these problems calls for a collaborative effort of all societal actors, be it governments, firms, NGOs, or citizens, in finding and implementing solutions. One of the goals formulated in the resolution is sustainable consumption and production. This highlights the responsibility of consumers and producers in the context of sustainability. However, consumers and producers alike face an information gap: the question ’how sustainable is a product?’ is not easily answered. But gaining the relevant information will enable them to make informed decisions regarding a product’s sustainability characteristics. The work presented in this paper aims to solve this problem from an information systems (IS) perspective, whereby the two guiding research questions are:

1. How to assess a product’s sustainability?
2. How to visualize and communicate a product’s sustainability?

This paper proposes a design-oriented approach [9] that contributes to the field of enterprise modeling, by developing a modeling method [7] for product sustainability. The modeling method is part of an information system that enables researchers and practitioners to model and analyse product life cycles with respect to sustainability.

The following section 2 describes this general vision and research idea: the envisioned artifacts and the relationship to the research field of life cycle assessment is laid out. As a first step towards this goal, the author has developed a domain-specific modeling language (DSML) and software tool that is limited in scope, but already supports assessment and visualization of certain aspects of product sustainability. This initial step towards the research goal is described in section 3. Section 4 draws a conclusion and briefly discusses how the existing work can be extended towards the general vision.

2 A Design-Oriented Approach to Product Sustainability

Design-oriented IS research, as proposed by Österle et al. [9], means to develop and evaluate artifacts that provide utility to specific stakeholders by solving a specific problem. In this case, the focused stakeholder group consists of organizations, e.g. firms and NGOs, that want to analyze a product's sustainability. Organizations willing to disclose the gained knowledge shall further be supported in providing transparency to external stakeholders, like customers or suppliers.

As the most common definition of sustainability - to meet the needs of the present without compromising the ability of future generations to meet their own [11] - takes a global perspective, a product in itself can never be sustainable in this global sense. However, its production and use can have various levels of effects on global sustainability (see [6]). So the goal is to create visibility of social and environmental effects within a product life cycle (i.e. from the extraction of raw materials to production, use and finally disposal or recycling). Taking as example a modern ICT product like a smartphone or a laptop, potential sustainability problems can be identified along the whole life cycle: from the mining of so called "conflict minerals" that finance armed conflicts in Central Africa [2] to the illegal shipping of electronic waste to developing countries, where the products are disassembled under hazardous conditions [1]. Regarding the domain of product sustainability, the life cycle assessment (LCA) research community provides methods for identifying and assessing these effects. While the "classic" LCA methodology is mainly concerned with environmental aspects, the inclusion of social aspects has recently grown more important [10].

Hereby, the main research idea is to design a modeling method for product sustainability that incorporates a LCA perspective. A modeling method consists of a modeling language, a modeling procedure, and supporting mechanisms and algorithms [7]. These elements can then be implemented in an information system that supports the assessment of product sustainability to allow for informed decisions. The user roles of modeler and viewer have differing needs concerning such a system. A viewer might only be interested in aggregated visualizations and results based on the created models. But, in order to improve the quality of the results, the models need to be as detailed as possible. This motivates the need for specific mechanisms and algorithms that perform calculations on the input models and transform detailed models into aggregated visualizations.

The envisioned modeling language and procedure, as well as the necessary mechanisms and algorithms are described in the following:

Modeling Language and Procedure: A meta-model of product sustainability forms the conceptual basis. It further describes and structures the problem domain. The general idea is then, to develop a DSML and procedure based on the meta model. The language and procedure will be implemented in a software tool that serves several purposes:

- To support a prospective modeler in structuring and collecting information about a product life cycle.
- To enable a viewer to explore and identify sustainability risks within the product life cycle.

Mechanisms and Algorithms: In order to provide aggregated results and perform assessments, the proposed system draws inspiration from LCA methods. Examples for computational analysis methods that were developed for (environmental) LCA make use of sequential methods, linear programming or petri nets [5]. Any of them may potentially be adapted for the proposed system. However, as the goal is to integrate environmental and social assessment, the methods need to be evaluated regarding their adaptability to social aspects. Apart from that, petri nets seem like an interesting choice, as this would (given corresponding transformation mechanisms) allow for the possibility to 'drill down' from aggregated models, using a domain-specific notation, to the underlying formal petri nets.

3 Initial Steps and Existing Work

As a first step towards the research goal outlined above, the author has developed a DSML named 'TracyML'. The modeling language serves the purpose of visualizing social risks within a product life cycle. In its current state, TracyML focuses on the early life cycle phases from extraction of raw material to the assembly of the final product. It is further restricted to the stakeholder group workers and visualizes social aspects like the risk of forced labour or excessive working hours. The language was developed by adapting a method proposed by Frank [4]. It features a meta-model and a graphical notation, which was developed with an emphasis of usability and understandability.

Additionally, a web-based prototype modeling tool was developed. A running instance and the source code is available under www.gotracy.org. It implements the features of TracyML, and further visualizations based on user-created models. As part of the evaluation, the software tool was applied to two ICT product use cases, a computer mouse (www.nager-it.de) and solder wire (www.fairloetet.de). In both cases, the organizations provided supply chain data that could be used to model the extraction of raw materials and assembly of components. The assessment of the social risks is then based on generic regional data, taking into account the location of the known production sites in the supply chain.

4 Conclusion and Outlook

The next step is to extend the TracyML meta model to a broader sustainability perspective, covering social and environmental aspects. The preceding work has laid the basis for the envisioned information system, but already exposed some challenges. For example, in order to not overload the user, the complexity of a modeling language should be kept in check [8]. But, as system visibility is an important enabler for sustainability [3], the underlying assumptions and calculations should be traceable for the interested user. Here, the developed solution needs to strike a balance between simplicity and traceability, which needs to be continuously evaluated. Finally, the work presented here contributes to enterprise modeling by addressing the arising challenge of product sustainability. The results provide utility for firms, NGOs, and consumers. Prospective users are supported in identifying sustainability issues in a product life cycle, and deducting courses of action through informed decision.

References

1. BBC: Making a living from toxic electronic waste in Ghana (2014), http://www.bbc.com/news/technology-26239741
2. BBC: US firms 'fail' in conflict mineral disclosures - report (2015), http://www.bbc.com/news/business-32403315
3. Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., Venters, C.C.: Sustainability Design and Software: The Karlskrona Manifesto. In: Proceedings of the 37th International Conference on Software Engineering-Volume 2. pp. 467–476. IEEE Press (2015)
4. Frank, U.: Some Guidelines for the Conception of Domain-Specific Modelling Languages. In: EMISA. vol. 190, pp. 93–106 (2011)
5. Heijungs, R., Suh, S.: The Computational Structure of Life Cycle Assessment, vol. 11. Springer Science & Business Media (2013)
6. Hilty, L.M., Aebischer, B.: ICT for sustainability: An emerging research field. In: ICT Innovations for Sustainability, pp. 3–36. Springer (2015)
7. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Proceedings of the Third International Conference EC-Web. vol. 2455, p. 182. Springer (2002)
8. Moody, D.: The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35(6), 756–779 (2009)
9. Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J.: Memorandum on design-oriented information systems research. European Journal of Information Systems 20(1), 7–10 (2011)
10. UNEP/SETAC Life Cycle Initiative: Guidelines for Social Life Cycle Assessment of Products. United Nations Environment Programme (2009)
11. United Nations General Assembly: Our Common Future - Brundtland Report (1987)
12. United Nations General Assembly: Resolution 70/1 Transforming our world: The 2030 Agenda for Sustainable Development (2015)

All links were last followed on January 15, 2017.

An Analysis of Metaheuristic to SLA Establishment in Cloud Computing

Leonildoi J. M. Azevedo[1], Julio C. Estrella[1], Claudio F. M. Toledo[1] and Stephan Reiff-Marganiec[2]

[1] Institute of Mathematics and Computer Sciences, University of Sao Paulo, SP, Brazil
`leonildo.azevedo@usp.br, {jcezar, claudio }@icmc.usp.br`
[2] University of Leicester, Leicester, UK
`srm13@leicester.ac.uk`

Abstract. Nowadays the access to a cloud computing environment is provided on-demand offering transparent services to customers. Although the cloud allows an abstraction of the behavior of the service providers in the infrastructure (involving logical and physical resources), it remains a challenge to fully comply with the Service Level Agreements (SLAs), because, depending on the service demand and system configuration, the providers may not be able to meet the requirements of the customers. This paper proposes algorithms to address the need for optimization when handling computational resources before establishment the SLA.

Keywords: Cloud computer, SLA, optimization, IaaS

1 Introduction

In recent years, cloud computing has been one of the most widely discussed areas of Information Technology (IT). Cloud computing can also be regarded as an extension of other paradigms such as standard grade and utilitarian computing; this enables business applications to be viewed as sophisticated services which can be accessed by means of a network (the Internet) [5].

Cloud Computing is conceptually divided between three basic services models: **Software as a Service (SaaS)**, **Platform as a Service (PaaS)**, and **Infrastructure as a Service (IaaS)**. These models set out the capacities of the cloud services and how such services can be rendered to clients. This approach can be viewed as a division of rendered services in abstract layers [5].

Cloud Computing is a vast and complex computing paradigm that is closely bound up with the business dealings of its clients. It has always been a challenging task to provide clients with a suitable infrastructure for rendering services. Different clients require different resources depending on their hosted applications or demands made by users for these applications. For example, a particular client may have a CPU-Bound application while another client has a Memory-Bound application. One client might have an application with a large number of simultaneous accesses, whereas another client might have an application with only a few random accesses.

The Cloud providers generally offer several configurations of virtualized resources (combinations of CPU, Memory and Disc space based on Virtual Machines (VMs)). Some configurations of virtual machines are predefined. Providers such as Amazon EC2 and Microsoft Azure employ a methodology in which the clients themselves are responsible for giving a precise estimate of the necessary resources and selecting the VM to be contracted [8].

It should be remembered that the users do not always have the technical knowledge to handle the provisioning of resources, not to mention the fact that this task can be burdensome for them. Therefore, the application of some kind of optimization technique for the provisioning of resources can make this an automated task. This can ensure the maximum use of computational resources and hence, the user need not to pay for more than she/he really needs. They will pay a fair price for the contracted service.

In this paper, our concern is with the IaaS service model and thus we have set out to create and evaluate optimization algorithms that fully comply with SLAs. We propose two optimization algorithms aimed at overcome the highlighted challenges. First, we describe a deterministic algorithm that is able to search exhaustively the problem solution space. Next, a micro-Genetic Algorithm (μGA) is proposed aiming to search more efficiently the solution space.

The paper presents two algorithms providing novel solutions for automating resource allocation in cloud computing and initial results of their performance.

2 Literature Review

The cloud is a highly scalable environment in which the demand for services can change unexpectedly. Thus, the automatic allocation of resources to meet this demand is becoming an issue of great interest in both the academic and industrial world.

Correct provisioning allows for a better use of available computational resources and the whole infrastructure which comprises cloud, since it carries out a more efficient mapping between the loading and resources [2]. For example, the task of providing more computational resources to a client becomes more feasible and it can be made available more easily since the resources are virtualized. Moreover, while from the standpoint of the clients, the resources can be regarded as unlimited, it should be remembered that the allocation of more resources has an influence on the final cost. This cost has to be passed on to the client and it requires effective mechanisms on the the provider side.

There are several studies that analyze mechanisms for managing resources in a cloud environment. The suggestion of Amazon, that there should be an automatic reconfiguration of the infrastructure of the clients, is based on monitoring by means of Cloud-Watch Alarms and Scaling Policies. The scaling of Amazon Web Services (AWS) can be carried out by employing metrics which are generally based on the consumption of CPU time or on policies that are essentially divided between Manual Scaling, Dynamic Scaling and Scheduled Scaling [1].

There are many complex problems within the area of cloud computing that can be addressed by solutions that are found through optimization. For example, a) the problems of allocating virtual machines in a real machine [10]; b) energy saving [3]; c) scaling and load balancing of applications in virtual machines [9]; d) the use of resources aimed at reducing costs, while still guaranteeing a satisfactory performance [6]; and e) ensuring the QoS is in compliance with the SLA.

All these problems require optimal or sub-optimal solutions which can be achieved through techniques that are conceptualized within the area of optimization. This study seeks to tackle the problem of allocating resources in a suitable way by employing a heuristic and a meta-heuristic.

3 Methods

Some experiments were conducted with the QoS parameters described earlier in Section 1. These experiments were carried out in the CloudSim Simulator 3.0.3 3 version [3], with the aid of a computer with an AMD Phenom(tm) II X6 1090T Processor, with 16 GB of RAM memory, 1.5 TB of disc storage and the Ubuntu 14.04.3 LTS operational system with a kernel version 3.13.0. CloudSim was configured for the execution of three types of VM requests based on ***m3.medium, m3.large* and *m3.xlarge*** from Amazon EC2[4].

When the experiments were conducted, an SLA was stipulated for a particular client, the QoS attributes consisting of Capacity (C^c), Time (T^c), Availability (A^c) and Cost/hour (C/h^c). The simulation involves executing the client application in the capacity described in the SLA. As a result of the execution, the response time, availability and cost/h returned by CloudSim are obtained. If the parameters returned by the CloudSim are not compatible with the descriptions in the SLA, a meta-heuristic is applied to find a solution (Capacity (C^*), Time (T^*), Availability (A^*) of cost/h (C/h^*)) which can be best adapted to the requirements of the client. Thus, the proposed method will try to define for the provider the best arrangement of VMs ($s*, m*, l*$).

Two algorithms were designed to solve this problem: a deterministic algorithm (vide Algorithm 1) and a μGA (vide Algorithm 2). These methods will optimize the number of virtual machines contracted by the client. Thus, the representation of the solution (encoding) is defined by (s, m, l), which are the number of Virtual Machines (VMs) of type Small, Medium and Large, respectively, that will better satisfy client requests.

The representation of solutions (s, m, l) are evaluated by CloudSim Simulator 3.0.3 3 version, that is configured for the execution of three types of VMs. As a result of its execution, the capacity (C^*), response time (T^*), availability (A^*) and cost/h (C/h^*) returned by CloudSim are obtained. If these parameter values are not compatible with those defined in the SLA (C^c, T^c, A^c, C/h^c), another representation of solution (s', m', l') must be evaluated. This compatibility is estimated by Equation(1).

[3] http://www.cloudbus.org/cloudsim/

[4] https://aws.amazon.com/pt/ec2/instance-types/

$$f(P[i]) = |\frac{C^c - C^*}{C^*}| + |\frac{T^c - T^*}{T^*}| + |\frac{A^c - A^*}{A^*}| + |\frac{C/h^c - C/h^*}{C/h^*}| \tag{1}$$

The Manhattan Distance [4] is employed to estimate how close the solution is to the values stipulated in the SLA. There are four objectives with scale of values, so a standardized system is necessary based on Gap values.

The search space is defined from the number of machines adjusted by the client, where an amount 50% above or below the desired number of VMs are evaluated. For example, if the client contracted 50 VMs, the possible interval to explore remains between 25 and 75 VMs. In this case, representation of solutions as (s, m, l)=$(1, 20, 4)$ or (s, m, l)=$(0, 0, 75)$ are allowed since $25 \leq (s+m+l) \leq 75$. Thus, all possible combinations of VMs within this range of 50% is evaluated aiming to obtain the best value for Equation (1).

Algorithm 1 describes the deterministic algorithm, where all possible combinations for (s, m, l) are exhaustively evaluated and the best one is returned. Algorithm 2 shows the proposed μGA. This method is a genetic algorithm (GA) version which operates with a smaller-sized population and employs a convergence criterion that allows reinitialization.

Algorithm 1: Deterministic algorithm

```
Input: SLA: $C^c, T^c, A^c, C/h^c$
1 //2. Sweep all the search space
2 repeat
3     //generate a capacity to be evaluated
4     Generate($s, m, l$)
5     Evaluate($s, m, l$)
6 until Until all $(s, m, l)$ possibilities within the interval have been tested;
7 //return a better configuration found to satisfy the SLA of the client
8 return $SLA^* : C^*, T^*, A^*, C/h^*$
```

Algorithm 2: μGA algorithm

```
Input: SLA: $C^c, T^c, A^c, C/h^c$
1 //1. Generate a population with 5 individuals
2 InitializePopulation(P)
3 //Assess the fitness of each individual in the population
4 Evalutate(P)
5 repeat
6     Selection(P)
7     Crossover(P)
8     Mutation(P)
9     Evaluate(P)
10    if the best individual is not updated after 10 iterations then
11        Reinitialize(P)
12    end
13 until time limit has been reached;
14 //return the best individual
15 return $SLA^* : C^*, T^*, A^*, C/h^*$
```

The population was adjusted until it settled at only 5 individuals, each of which represents a possible VM configuration (s, m, l) for the client. The initialization generantes randomly individuals (s, m, l) such as $min \leq (s + m + l) \leq max$, where the possible range is defined by $[min, max]$.

A tournament is applied to select between two individuals one to take part in crossover. The BlX-α crossover [7] is applied to generate new individuals. However, if $(s + m + l) \leq min$ or $max \leq (s + m + l)$ for the new individual, the necessary amount is added or deduced from the last position (l). If it is not enough, the amount can be also reduced from position m.

4 Computational Results

The computational tests were divided into 10 scenarios compounded by different number of VMs available in a data center as shown in Table 1. We report the range to explore solutions $[min, max]$, the Execution Time spent and the total Number of combination evaluated by the deterministic algorithm for each scenario.

For a data center with 10 VMs, it was assumed that the client requested 5 VMS, so the range to be explored is $[2, 8]$ in this case. It can be observed that for 20 VMs, a fairly high execution time is already obtained in a scale of minutes (2.19 minutes). The time increases exponentially for the others scenarios as a result of the large number of VMs. The execution takes longer than 20 minutes for $VMs > 50$. The deterministic algorithm is able to find the optimal solution for this problem once it explores completely the solution space. However, these results indicates that the method is not viable for practical proposes even for a small number of VMs.

The μGA was executed 100 times for each scenario with 3 minutes as stop criterion by execution. It was assumed that 3 minutes is a reasonable time to reply a client request. After all executions, the success rate achieved by μGA was estimated by comparing the best solution of μGA with the optimal solution found by the deterministic algorithm. A sucess is computed if μGA finds the optimal solution in some run. The proposed μGA was able to find the optimal solutions for all scenarios in all 1000 executions. Table **??** reports the average Time with standard deviations to find the optimal solution.

This correctness rate was obtained in less than 1 minute for all scenarios by μGA, in contrast with the deterministic method that took more than 1 minute in all cases, except by VMs=10.

Table 1. Response time to the μGA and Deterministic model of 10 scenarios scenario from 10 until 100 available VMs.

Number of VMs	$[min, max]$	**Execution Time (minutes)**	**Time μGA (minutes)**
10	$[2, 8]$	0.35	0.13 ± 0.15
20	$[5, 15]$	2.19	0.54 ± 0.32
30	$[7, 22]$	6.28	0.19 ± 0.19
40	$[10, 30]$	14.83	0.32 ± 0.21
50	$[12, 37]$	27.08	0.56 ± 0.24
60	$[15, 45]$	47.53	0.17 ± 0.19
70	$[17, 52]$	72.38	0.14 ± 0.17
80	$[20, 60]$	108.56	0.17 ± 0.20
90	$[22, 67]$	150.25	0.19 ± 0.21
100	$[25, 75]$	213.86	0.16 ± 0.16

5 Conclusion

It is not a trivial task to provide a cloud computing user with an efficient infrastructure, that respects the SLA and its QoS attributes, while at the same time seeking to reduce costs. Within the domain of cloud computing, the most wide-ranging problems can be mapped out in solutions that generally involve optimization based on their complexity and the large number of resources that can be scalable.

This article addresses one of these challenges which was to provide the client with a self-manageable infrastructure with the provision of SLA agreed between the client and provider. Our proposal mapped some of the QoS attributes that determine the criteria for the SLA and we also designed and analyzed algorithms that allow an optimized reconfiguration of the infrastructure based on these criteria. The results provide evidence that the μGA algorithm is efficient and applicable to the solution of the problem.

In future work, we intend to carry out new tests with a bigger number of VMs and clients. We are also going to design new meta-heuristic algorithms so that a comparison can be made between them, and at the monitoring phase. Other QoS attributes will be added to the SLA and new constraints to the problem such as defining a maximum cost threshold that the client is able to afford.

Acknowledgments The authors thank CNPq, CAPES and FAPESP (processes IDs: 16/14219-2 and 15/11623-4) for the financial support funding the this research work.

References

1. Amazon: Scaling the size of your auto scaling group (2015), available at: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/scaling_plan.html. Last Access: 01/20/2015
2. Batista, B.G., Estrella, J.C., Ferreira, C.H.G., Leite Filho, D.M., Nakamura, L.H.V., Reiff-Marganiec, S., Santana, M.J., Santana, R.H.C.: Performance evaluation of resource management in cloud computing environments. PloS one 10(11), 21 (2015)
3. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28(5), 755 – 768 (2012), http://www.sciencedirect.com/science/article/pii/S0167739X11000689, special Section: Energy efficiency in large-scale distributed systems
4. Black, P.E.: Manhattan distance. Dictionary of Algorithms and Data Structures 18, 2012 (2006)
5. Buyya, R., Broberg, J., Goscinski, A.M.: Cloud Computing Principles and Paradigms. Wiley Publishing (2011)
6. Byun, E.K., Kee, Y.S., Kim, J.S., Maeng, S.: Cost optimized provisioning of elastic resources for application workflows. Future Generation Computer Systems 27(8), 1011 – 1026 (2011), http://www.sciencedirect.com/science/article/pii/S0167739X11000744
7. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Natural Computing Series, Springer Berlin Heidelberg (2007), https://books.google.com.br/books?id=7IOE5VIpFpwC
8. Huu, T.T., Montagnat, J.: Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. pp. 612–617. IEEE (2010)
9. L.D., D.B., Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing 13(5), 2292 – 2303 (2013), http://www.sciencedirect.com/science/article/pii/S1568494613000446
10. Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. Journal of Network and Computer Applications pp. – (2016), http://www.sciencedirect.com/science/article/pii/S1084804516000291

PROtEUS++: A Self-managed IoT Workflow Engine with Dynamic Service Discovery

Ronny Seiger, Steffen Huber, and Peter Heisig

Institute of Software and Multimedia Technology,
Technische Universität Dresden, D-01062, Germany
{Ronny.Seiger, Steffen.Huber, Peter.Heisig}@tu-dresden.de

Abstract. Despite offering various advantages, the usage of Business Process Management technologies to orchestrate workflows in the Internet of Things (IoT) is still in its infancy. In this work, we demonstrate an extended version of our PROtEUS process execution system for IoT. Besides the processing of sensor streams and interacting with humans, PROtEUS++ is capable of dynamic service invocation as well as self-management to detect and repair errors that happen during process execution. We show the system executing various dynamic and error-prone processes in the Smart Home.

1 Introduction

The IoT and its associated technologies are currently transforming the world of purely digital information systems into Cyber-physical Systems (CPS). The interaction with the physical world requires feedback loops and flexible service composition to compensate errors, which makes selection of IoT services highly context dependant. The BPM and SOA communities provide promising technologies for increasing the automation within CPS. However, the convergence of these research areas has only just started. With PROtEUS++ we present a novel self-managed IoT/CPS workflow engine with dynamic service discovery, which combines BPM and SOA concepts applied in the context of CPS. We show new use cases for PROtEUS++ in the smart home providing an adaptive light control system and executing distributed processes on mobile robots.

2 PROtEUS Base System

The *PROtEUS* base system and its highlighted extensions are shown in Fig. 1. PROtEUS is a workflow system designed to execute processes in the IoT [4]. It features a Petri net based core engine. A complex event processing (CEP) engine is used to process sensor streams from digital and physical sources. The service invoker is able to call a variety of external Web services or services deployed on its local service platform and thereby invoke physical actuators. Users can interact with processes via the HumanTask Handler and the Process Manager to control the execution. The Distribution Manager allows for executing subprocesses on other remote PROtEUS instances. The extended *PROtEUS++* system adds the *Semantic Access Layer* and the *Feedback Service* to the base system.

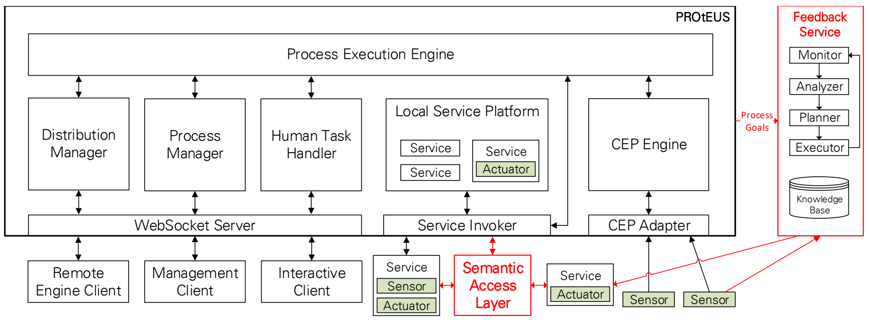


Fig. 1. The basic process execution system (PROtEUS) and its extensions.

3 Semantic Service Selection

PROtEUS is extended with a *Semantic Access Layer* (SAL) [2] as shown in Fig. 2. The core of this service is a knowledge base that contains information about all IoT sensors and actuators, their capabilities and contexts, as well as their associated IoT services and interfaces. User defined SPARQL queries can be sent to the SAL from a process activity to find and invoke IoT services in a specified context at runtime. If an IoT service can be found as a process resource, a call to our IoT middleware (here: *OpenHAB*) is issued to execute this service.

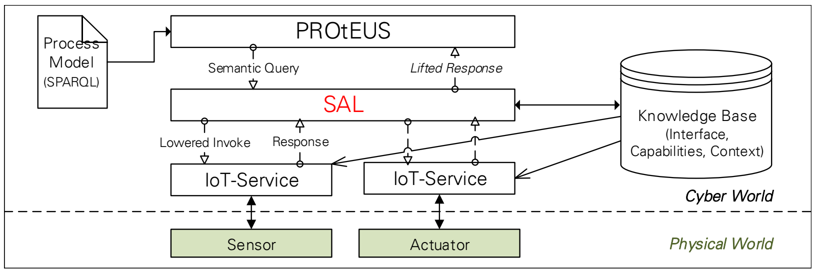


Fig. 2. Architecture of the semantic access layer (SAL).

4 Feedback Service for Self-management

The base system can be used in combination with the *Feedback Service* [3], which adds self-management (here: *self-healing*) in the form a cyber-physical feedback loop to PROtEUS (cf. Fig. 1). This service implements the *MAPE-K* loop for autonomous systems applied to the process execution [1]. This loop enables the linking of the execution of process activities to their physical effects: *Monitoring* gathers relevant sensor data from the environment and execution system; *Analysis* analyzes the data regarding the fulfillment of the process goals; *Planning* searches for compensations (here: alternative actuators or services) in case of unexpected errors; and *Execution* executes these compensations. All components use the *Knowledge Base* to store and retrieve relevant information (e. g., for finding replacement services). This loop is executed until a process goal is reached or cancelled if errors cannot be compensated. These *Process Goals* are specified on the activity, subprocess or process level. They contain the paths to relevant sensor data, a condition defining the successful execution, and a condition defining the need for entering the Planning phase due to an error [4].

5 Demo

We present several real-world use cases of PROtEUS++ executing three example processes. 1) A process demonstrating the base system and the dynamic service selection via the SAL–a health monitoring process asking the user for its well-being in case of a detected emergency and calling an ambulance if the user is unresponsive. 2) A process demonstrating the base system interacting with the Feedback Service. A continuous process controls the light levels in a room to be within certain thresholds–selecting an alternative light source or notifying the user if the lights fail. 3) A process demonstrating the execution and feedback control of a distributed process on a service robot–repeating the subprocess on another robot if it fails during the driving to different locations. Along with these processes, we present the modelling tool [5] as well as a mobile control center app to interact with the workflows [6].

Acknowledgements

This research has received funding under the grant number 100268299 by the European Social Fund (ESF) and the German Federal State of Saxony.

References

1. Autonomic Computing: An architectural blueprint for autonomic computing. IBM Publication (2003)
2. Huber, S., Seiger, R., Schlegel, T.: Using semantic queries to enable dynamic service invocation for processes in the internet of things. In: ICSC. pp. 214–221 (Feb 2016)
3. Seiger, R., Huber, S., Heisig, P., Assmann, U.: Enabling Self-adaptive Workflows for Cyber-physical Systems, pp. 3–17. Springer International Publishing (2016)
4. Seiger, R., Huber, S., Schlegel, T.: Proteus: An integrated system for process execution in cyber-physical systems. In: Enterprise, Business-Process and Information Systems Modeling, LNBIP, vol. 214, pp. 265–280 (2015)
5. Seiger, R., Keller, C., Niebling, F., Schlegel, T.: Modelling complex and flexible processes for smart cyber-physical environments. JOCS 10, 137–148 (2015)
6. Seiger, R., Lemme, D., Struwe, S., Schlegel, T.: An interactive mobile control center for cyber-physical systems. In: UbiComp Adjunct Proceedings. pp. 193–196. UbiComp '16, ACM, New York, NY, USA (2016)

Author Index