SERVICE COMPUTATION 2014

The Sixth International Conferences on Advanced Service Computing

ISBN: 978-1-61208-337-7

May 25 - 29, 2014

Venice, Italy

SERVICE COMPUTATION 2014 Editors

Arne Koschel, Hochschule Hannover, Germany

Alfred Zimmermann, Reutlingen University, Germany

SERVICE COMPUTATION 2014

Foreword

The Sixth International Conferences on Advanced Service Computing (SERVICE COMPUTATION 2014), held between May 25-29, 2014 in Venice, Italy, targeted service computation on different facets. It considered their ubiquity and pervasiveness, WEB services, and particular categories of day-to-day services, such as public, utility, entertainment and business.

We take here the opportunity to warmly thank all the members of the SERVICE COMPUTATION 2014 Technical Program Committee, as well as all of the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to SERVICE COMPUTATION 2014. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the SERVICE COMPUTATION 2014 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that SERVICE COMPUTATION 2014 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the area of advanced service computing.

We are convinced that the participants found the event useful and communications very open. We hope that Venice, Italy, provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

SERVICE COMPUTATION 2014 Chairs:

Mihhail Matskin, KTH, Sweden
Hideyasu Sasaki, Ritsumeikan University - Kyoto, Japan
Bernhard Hollunder, Hochschule Furtwangen University – Furtwangen, Germany
Paul Humphreys, Ulster Business School/University of Ulster, UK
Arne Koschel, Hochschule Hannover, Germany
Michele Ruta, Politecnico di Bari, Italy
Alfred Zimmermann, Reutlingen University, German
Aida Omerovic, SINTEF, Norway
Martin Wynn, University of Gloucestershire, UK
Annett Laube, Bern University of Applied Sciences (BUAS), Switzerland
Claus Pahl, Dublin City University, Ireland
Ali Beklen, CloudArena, Turkey
Mark Yampolskiy, Valderbilt University, USA
Steffen Fries, Siemens Corporate Technology - Munich, Germany
Emmanuel Bertin, Orange Labs, France
Matthias Olzmann, noventum consulting GmbH - Münster, Germany
Sergey Boldyrev, HERE Espoo, Finland
Rong N. Chang, IBM T.J. Watson Research Center, USA
Wasif Gilani, SAP Research, UK

Alexander Kipp, Robert Bosch GmbH, Germany
Marcello Coppola, ST Microelectronics - Grenoble, France
Jan Porekar, SETCCE, Slovenia

SERVICE COMPUTATION 2014

Committee

SERVICE COMPUTATION Advisory Chairs

Mihhail Matskin, KTH, Sweden
Hideyasu Sasaki, Ritsumeikan University - Kyoto, Japan
Bernhard Hollunder, Hochschule Furtwangen University – Furtwangen, Germany
Paul Humphreys, Ulster Business School/University of Ulster, UK
Arne Koschel, Hochschule Hannover, Germany
Michele Ruta, Politecnico di Bari, Italy
Alfred Zimmermann, Reutlingen University, German
Aida Omerovic, SINTEF, Norway
Martin Wynn, University of Gloucestershire, UK
Annett Laube, Bern University of Applied Sciences (BUAS), Switzerland
Claus Pahl, Dublin City University, Ireland

SERVICE COMPUTATION 2014 Industry/Research Chairs

Ali Beklen, CloudArena, Turkey
Mark Yampolskiy, Valderbilt University, USA
Steffen Fries, Siemens Corporate Technology - Munich, Germany
Emmanuel Bertin, Orange Labs, France
Matthias Olzmann, noventum consulting GmbH - Münster, Germany
Sergey Boldyrev, HERE Espoo, Finland
Rong N. Chang, IBM T.J. Watson Research Center, USA
Wasif Gilani, SAP Research, UK
Alexander Kipp, Robert Bosch GmbH, Germany
Marcello Coppola, ST Microelectronics - Grenoble, France
Jan Porekar, SETCCE, Slovenia

SERVICE COMPUTATION 2014 Technical Program Committee

Witold Abramowicz ,Poznan University of Economics, Poland
Saeed Aghaee, University of Lugano, Switzerland
Riyad Alshammari, KSAU-HS University, Saudi Arabia
Dimosthenis S. Anagnostopoulos, Harokopio University of Athens, Greece
Julian Andrés Zúñiga, University of Cauca, Colombia
Ismailcem Budak Arpinar, University of Georgia, USA
Johnnes Arreymbi, School of Architecture, Computing and Engineering - University of East London, UK
Irina Astrova, Tallinn University of Technology, Estonia
Jocelyn Aubert, Public Research Centre Henri Tudor, Luxembourg
Benjamin Aziz, School of Computing - University of Portsmouth, UK
Youcef Baghdadi, Department of Computer Science - Sultan Qaboos University, Oman
Ebrahim Bagheri, Ryerson University, Canada
Zubair Ahmed Baig, KFUPM - Dhahran, Saudi Arabia

Victor Govindaswamy, Texas A&M University-Texarkana, USA
Mohamed Graiet, Institut Supérieur d'Informatique et de Mathématique de Monastir, Tunisie
Maki K. Habib, American University in Cairo, Egypt
Ileana Hamburg, IAT - Westfälische Hochschule Gelsenkirchen, Germany
Takahiro Hara, Osaka University, Japan
Sven Hartmann, Clausthal University of Technology, Germany
Martin Henkel, Department of Computer and Systems Sciences – Stockholm University, Sweden
Bernhard Hollunder, Hochschule Furtwangen University - Furtwangen, Germany
Wladyslaw Homenda, Warsaw University of Technology, Poland
Tzung-Pei Hong, National University of Kaohsiung, Taiwan
Samuelson W. Hong, Zhejiang University of Finance & Economics, China
Sun-Yuan Hsieh, National Cheng Kung University, Taiwan
Marc-Philippe Huget, LISTIC/Polytech Annecy Chambery/University of Savoie, France
Paul Humphreys, Ulster Business School/University of Ulster, UK
Mirjana Ivanovic, University of Novi Sad, Serbia
Hemant Jain, University of Wisconsin- Milwaukee, USA
Jinlei Jiang, Tsinghua University - Beijing, China
Ivan Jelinek, Faculty of Electrical Engineering - Czech Technical University Department of Computer Science and Engineering, Czech Republic
Alexander Jungmann, University of Paderborn, Germany
Alexandros Kaloxylos, University of Peloponnese, Greece
Rajaraman Kanagasabai, Institute for Infocomm Research, Singapore
Tahar Kechadi, University College Dublin, Ireland
Nhien An Le Khac, University College Dublin, Ireland
Hyunsung Kim, Kyungil University, Korea
Alexander Kipp, Robert Bosch GmbH, Germany
Manuele Kirsch Pinheiro, Université Paris 1 - Panthéon Sorbonne, France
Mourad Kmimech, l'Institut Supérieur d'informatique de Mahdia (ISIMA), Tunisia
Arne Koschel, Hochschule Hannover, Germany
Yousri Kouki, ASCOLA - INRIA, France
Natalia Kryvinska, University of Vienna, Austria
Patricia Lago, VU University Amsterdam, Netherlands
Ulrich Lampe, Technische Universität Darmstadt, Germany
Annett Laube-Rosenpflanzer, Bern University of Applied Sciences - Biel/Bienne, Switzerland
Guanling Lee, National Dong Hwa University, Taiwan
Keqin Li, SAP Product Security Research, France
Kuan-Ching Li, Providence University, Taiwan
Noura Limam, University of Waterloo, Canada
Cho-Chin Lin, National Ilan University,Taiwan
Damon Shing-Min Liu, National Chung Cheng University, Taiwan
Qing Liu, The Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia
Shih-His (Alex) Liu, California State University - Fresno, USA
Hui Ma, Victoria University of Wellington, New Zealand
Khaled Mahbub, City University, UK
Sabri A. Mahmoud, King Fahd University of Petroleum and Minerals, Saudi Arabia
Kurt Maly, Old Dominion University, USA
Lefteris Mamatas, University College London, UK
Gregorio Martinez, University of Murcia, Spain

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Decentralized and Reliable Orchestration of Open Services

Abul Ahsan Md Mahmudul Haque and Weihai Yu
University of Tromsø – The Arctic University of Norway
Email: {Mahmudul.Haque, Weihai.Yu}@uit.no

***Abstract*—An ever-increasing number of clouds and web applications are providing open services to a wide range of applications. Whilst traditional centralized approaches to services orchestration are successful for enterprise service-oriented systems, they are subject to serious limitations for orchestrating the wider range of open services. Decentralized orchestration provides an attractive option for applications based on open services. However, decentralized approaches are themselves faced with a number of challenges, including the possibility of loss of dynamic run-time states that are spread over the distributed environment. This paper presents a dynamic replication scheme for a decentralized approach to orchestration of open services, where a network of agents collectively orchestrate open services using continuation-passing messaging.**

Keywords-web service; peer-to-peer; replication.

I. INTRODUCTION

An increasing number of individuals and enterprises are having an increasing number of their data and business functionality on line and in the cloud. To enable new applications to access these data and functionality, cloud providers and online business applications are offering open services through published Application Program Interfaces (APIs). Service orchestration is the coordination and conduct of the execution of multiple open services in the new applications [1].

Two technologies are highly relevant to the support of applications built on top of open services. (1) Web mashups are web applications that use content from multiple open services. ProgrammableWeb (www.programmableweb.com), for instance, lists thousands of open services and mashups. Although web mashups have been around for several years, they are still very limited in functionality (i.e., content only) and systematic support. Most noticeably, they are typically hand-crafted with low-level programming details. Execution of external open services are conducted by the web servers running the mashups. (2) Service-oriented computing (SOC) has been very well developed and supports most of the functionality such open-service based applications need. Traditionally, SOC focuses on cost-effective construction and integration of sophisticated applications within and across organizational boundaries. Therefore, unlike applications based on external open services, services composed in SOC generally limit themselves within enterprises or between enterprises with mutual agreements (this is generally known as services choreography [1]). Usually, dedicated central engines carry out the orchestration of composite services.

Recently, there have been efforts that bring the SOC technology to the cloud and open-service based applications. For example, Amazon SWF [2] allows applications to coordinate work (including service invocations) across distributed components.

In all current approaches, services are orchestrated either by dedicated central engines (SOC), or by sites hosting applications (mashups). This clearly has advantages, such as control and overview of global run-time status. However, application sites are typically vulnerable with respect to availability, scalability and reliability, whereas finding feasible central engines is hard when the services are beyond enterprise boundaries [3]. Even if such an engine exists (as with Amazon SWF), relying on central engines and/or individual big-name vendors would be subject to issues like censorship, surveillance, policy-dependence etc. [4]. Furthermore, because open services are potentially spread all over the world, long network delays are unavoidable when the locations of central engines are fixed.

Based on the above observations, we believe a decentralized or peer-to-peer approach to open-services orchestration would be an attractive option to a wide range of next generation open-service based applications. There have been research activities in the SOC community on decentralised orchestration of services (more on these in Section VII on related work). It is generally challenging to support reliable orchestration of external services that could be unreliable. It is even more challenging for decentralized orchestration over a large group of unreliable peers or agents.

Our decentralized orchestration mechanism is called *continuation-passing messaging* (CPM) [5][6]. Our earlier work addressed issues with exception handling and recovery in order to support reliable orchestration when external services are unreliable. In this paper, we present a dynamic replication scheme for reliable orchestration with potentially unreliable orchestration agents.

The paper is organized as the following. Section II gives a motivating example. Section III presents a peer-to-peer system model for services orchestration. Section IV reviews CPM. Section V presents replicated CPM, the main contribution of this paper. Section VI presents performance results. Section VII discusses related work. Section VIII concludes.

II. EXAMPLE

Consider an application that assigns reviewers to papers submitted to a conference for reviewing. The application achieves this by doing the following. It first uses digital library L to get a ranked list of candidate reviewers for each submitted paper based on the title and keywords of the paper as well as the publications of the candidate reviewers. Then, for the candidates above a certain threshold, it uses citation indexing service I to refine the shorted list based on co-authorship, affiliation and citations. Finally, it uses the refined list and its on-premise data, such as reviewers' interests, to assign the reviewers to the paper.

 ISBN: 978-1-61208-337-7

The application may handle different exceptions differently. If during execution the digital library becomes unavailable, it may try another digital library. If the citation indexing services becomes unavailable, it may simply accept the ranking list generated so far without any further refinement. It may be important for the conference organizer that once the application starts to run, it keeps running until the final assignments are done.

When the services of the example application are orchestrated by a central engine or by the site running the application, for every paper-reviewer pair, there is at least a round trip of messages between the engine and the service. Ideally, the engine can be placed close to the open service, and the placement can be done at run time. This is the basic idea behind continuation-passing messaging.

III. System Model

A *service provider* (*SP*) provides services through an open API with a number of operations. We use S_a, S_b etc., to denote SPs and a, a_1, a_2, $\bar{a}$ etc., to denote operations provided by S_a. Operation a of S_a is invoked with message $invoke(a)$ to S_a. A *service-based application* (*SA*) consists of invocations to a number of service operations in a given prescribed order. Without loss of generality in our study, we adopt a service-composition model similar to Web Services Business Process Execution Language (WS-BPEL) [7].

Fig. 1 shows an example SA p that consists of invocations to operations a at S_a, b at S_b, c at S_c and d at S_d. The SA first invokes a and then forks two parallel branches. The first branch invokes b n times in a loop. The second branch invokes c and d in sequence.

p: $scope($
  $sequence($
    $invoke(S_a, a, \bar{a})$,
    $fork($
      $loop(n, invoke(S_b, b, \bar{b}))$,
      $scope($
        $sequence($
          $invoke(S_c, c)$,
          $invoke(S_d, d, \bar{d}))$,
        any: $sequenc(compensate, invoke(S_e, e))))))$,
  any: $compensate)$

Figure 1. An example SA.

We assume that operations a, b and d have reverse operations $\bar{a}$, $\bar{b}$ and $\bar{d}$, and that operation c is read-only and does not need a reverse operation. The element $invoke(S_a, a, \bar{a})$ means: "run service operation a at S_a; if p has to be rolled back due to an exception that occurs after operation a successfully returns but before the entire p finishes, run service operation $\bar{a}$ to compensate for the executed effect of a". Notice that $invoke(S_a, a, \bar{a})$ is an SA construct that is not understood by S_a. S_a only understands either $invoke(a)$ or $invoke(\bar{a})$.

A *scope* is a unit of exception handling. Exception handlers are associated with scopes. When an exception of certain type is thrown in a scope, all current activities in the scope are stopped and the corresponding exception handler is executed. In Fig. 1, the top level scope has an exception handler for any type of exceptions. It runs a single operation $compensate$ that rolls back the current execution using the recovery plan that

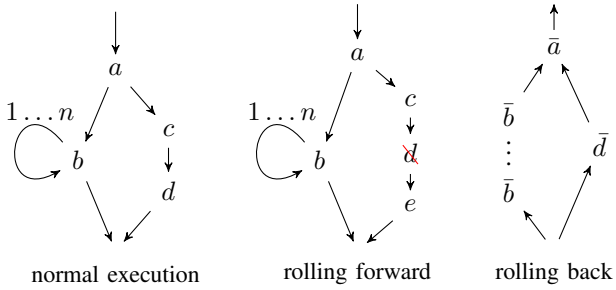


Figure 2. Control flow of example SA.

is automatically generated during the execution. The exception handler of the inner scope instead first rolls back the execution of the scope so far and then rolls forward by invoking an alternative service operation e.

Fig. 2 shows the control flows of a normal execution, a rolling forward (after the execution of d failed) and a rolling back (just before the entire p is about to finish).

Fig. 3 shows the service invocation messages (blue lines with arrows) when the SA is orchestrated by the site S_p that runs p. In this particular example, when the geographical distance between S_p and S_b is long, the loop may incur a long delay. If S_p is a mobile device, the execution of p could be costly and unreliable.

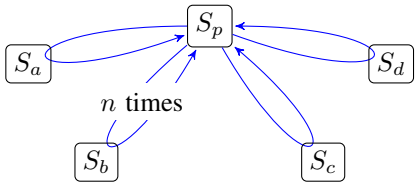


Figure 3. Centralized orchestration by the host SA server.

In our decentralized approach, a network of *orchestration agents* (OAs) collectively orchestrate the executions of SAs using *continuation-passing messaging* (CPM) [6]. We use A, A_a, A_b etc., to denote OAs. An OA has a *coverage* of SPs. Suppose a is an operation of S_a that is under the coverage of A_a. When a is part of an SA, invocation of a can be made via A_a.

At a specific moment, SPs may or may not be covered by OAs and OAs may have overlapping coverages. SPs become covered by OAs either by registration or through a learning process. An OA may not have the complete knowledge about the coverages of other OAs. At present, we assume that OAs learn effectively and every OA has nearly complete global knowledge of OA coverages.

An OA can run on a dedicated server, such as provided by a cloud provider. Alternatively, an SP may volunteer to be an OA as well. Being an OA may make its service more attractive. For example, if either S_b or the cloud hosting S_b has an OA, the loop in p may appear to be much more effective [5][6].

The basic tasks for the management of the OA network include OA membership, detection of OA availability, registration and discovery of SPs for their coverage, etc. Some of the tasks are already provided by existing software (such as the open source SERF [8]).

An SP may be unavailable, due to disconnection or system crash, and does not respond to invocations. An SP may also return an error. We assume that business critical services

 ISBN: 978-1-61208-337-7

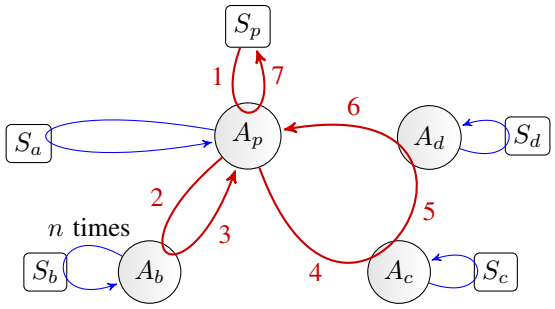
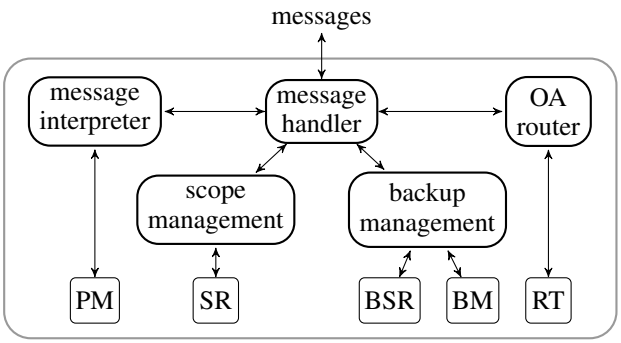


Figure 4. Messages with CPM Orchestration.



PM: pending messages, SR: scope registry, RT: routing table
BSR: backup scope registry, BM: Backup Messages

Figure 5. Structure of an Orchestration Agent.

support the at-most-once operation semantics. That is, an SP can recognize duplicated invocations and execute the same invocation at most once.

When an SP is not available or returns an error message, an exception is thrown so that an appropriate exception handler of the SA will handle it, such as by invoking an alternative service or rolling back the execution so far. Our orchestration mechanism guarantees effective propagation and handling of exceptions.

An OA may become unavailable in two ways. It may leave the OA network intentionally, or it may crash or get disconnected due to network failures. We assume a fail-stop crash model. The replicated CPM enhances the availability of the orchestration when the OAs are subject to such unavailability.

IV. CPM Overview

Fig. 4 shows the messages for CPM orchestration of the example SA p. Here we assume that SPs S_b, S_c and S_d are covered by OAs A_b, A_c and A_d, and S_a is not covered by any OA. There are three types of messages for services orchestration: service invocation messages (blue lines), CPM messages (red lines) and scope management messages (not shown in the figure). Orthogonal to the messages for services orchestration, OAs exchange routing messages to update the routing and health status of other OAs [6].

During orchestration, information like activity execution order and SA-aware data is carried in CPM messages in terms of continuations and environments. A *continuation* is a stack of activities that will be carried out, beginning from the head of the stack. An *environment* contains information of activity status and SA-aware data. To facilitate exception handling, messages also contain *compensation continuations*, which are rollback plans automatically generated during SA execution.

The initial continuation and environment of a CPM message are generated when an OA starts to orchestrate an SA. The message is later on sent to subsequent OAs that independently interpret the messages and invoke the service operations of the appropriate SPs. New continuations and environments are generated based on the messages being interpreted, as well as the outcomes of the service executions.

Fig. 5 shows the overall structure of an OA. In an OA, a *message interpreter* interprets an incoming or local message according to the head activity of the continuation. The following may happen during the interpretation.

An initial SA is converted into a CPM message. OAs are assigned to the corresponding activity elements according to the information in the OA router. For our example SA, $orch(p, S_p)$ —orchestration of p from S_p as specified in Fig. 1— is converted to $orch^{A_p}(scope^{A_p}(sequence(invoke^{A_p}(S_a, a, \bar{a})\ldots)))$, where orchestration activities like $orch$ and $scope$ are assigned to OAs A_p etc. For the purpose of space and readability, in what follows, we use notations like $scope^{A_p}(-)$ to suppress the details of the $scope$ element.

In some cases, a message can be interpreted alone. For example, $orch^{A_p}(scope^{A_p}(sequence(-))$ is interpreted into $scope^{A_p}(sequence(-)) \cdot eorch^{A_p}(-)$, which in turn is interpreted into $sequence(-) \cdot eos^{A_p}(-) \cdot eorch^{A_p}(-)$. Here $eorch$ and eos stand for *end-of-orchestration* and *end-of-scope*.

In other cases, multiple messages must be available to be further interpreted, for example, when messages from multiple parallel branches join. In this case, the first arrived messages are put in the *pool of pending messages* (PM). They are further interpreted when all dependent messages are available.

The interpretation of a message or multiple messages may lead to one or more new messages. Some messages are further interpreted locally by the same OA, like the $orch^{A_p}(-)$ above, and some are sent to other OAs for further interpretation.

If the head element of the continuation is an invocation assigned to the OA, the OA sends an invocation to the SP and waits for the result by putting a $wait$ message in its PM. For example, interpreting message $invoke^{A_p}(S_a, a, \bar{a}) \cdot fork(-) \cdot eos^{A_p}(-) \cdot orch^{A_p}(-)$, A_p sends $invoke(a)$ to S_a and puts $wait^{A_p}(S_a, a, \bar{a}) \cdot fork(-) \cdot eos^{A_p}(-) \cdot orch^{A_p}(-)$ in its PM. The $wait$ message is further interpreted according to the outcome of the invocation.

An OA may also be a scope manager and maintains some status information of each branch in its *scope registry* (SR). A scope manager is notified with a scope management message when the orchestration of an enclosing branch moves to a new OA. For example, when a branch moves from A_p to A_b, the scope manager A_p is notified of the move.

Table I lists the continuations in the remote CPM messages as shown in Fig. 4. Continuations of intermediate local messages are not shown in the table. In the table, κ is a continuation segment that is common in several continuations.

A $join$ element joins multiple parallel branches into one. It has an identifier and a join condition. Here the join condition is simply the number of branches to be joined.

The eos element marks the end of a scope and encapsulates necessary information for exception handling. The general form of an eos element is $eos^A(id, \kappa, h_1, h_2\ldots)$, where A

TABLE I. CONTINUATIONS IN MESSAGES

Msg	Continuation
1	$orch(scope(-), S_p)$
2	$invoke^{A_b}(S_b, b, \bar{b})$ $\cdot\, loop(n-1, invoke^{A_b}(S_b, b, \bar{b})) \cdot \kappa$
3	κ
4	$invoke^{A_c}(S_c, c) \cdot invoke^{A_d}(S_d, d, \bar{d})$ $\cdot\, eos^{A_p}(-) \cdot \kappa$
5	$invoke^{A_d}(S_d, d, \bar{d}) \cdot eos^{A_p}(-) \cdot \kappa$
6	$eos^{A_p}(-) \cdot \kappa$
7	$eorch(S_p)$
κ	$join^{A_p}(id_j, 2) \cdot eos^{A_p}(-) \cdot eorch^{A_p}(S_p)$

is the scope manager, id is the unique identifier of the scope, κ is a compensation continuation, and h_1, h_2 etc., are exception handlers. The compensation continuation is the recovery plan of the scope automatically generated during the orchestration. Table II lists the compensation continuations in the eos elements of the remote CPM messages. Notice that Messages 4, 5 and 6 have two eos elements for the two nested scopes.

TABLE II. COMPENSATION CONTINUATIONS

Msg	Compensation continuation	
1	nil	
2	$\bar{\kappa}$	
3	$invoke^{A_b}(S_b, \bar{b}) \cdots invoke^{A_b}(S_b, \bar{b}) \cdot \bar{\kappa}$	
4	$\bar{\kappa}$	nil
5	$\bar{\kappa}$	nil
6	$\bar{\kappa}$	$invoke^{A_d}(S_d, \bar{d})$
7	nil	
$\bar{\kappa}$	$join^{A_p}(id_{j'}, 2) \cdot invoke^{A_p}(S_a, \bar{a})$ $\cdot\, eos^{A_p}(-) \cdot eorch^{A_p}(S_p)$	

When A_b catches an exception, it runs the corresponding exception handler in the enclosing eos element and at the same time notifies the scope manager A_p of the exception. A_p then propagate the exception to the other branch(es).

For a scope with a single branch, an exception is completely handled where it is caught. This is the case of the nested scope of p. If A_d catches an exception, it handles the exception without notifying the scope manager A_p.

V. REPLICATED CPM

With CPM, information about the orchestration is usually already spread among multiple OAs. This information, if carefully maintained and updated, could be used to handle occasional unavailability of OAs. This is the key idea behind replicated CPM.

A. Selection of backup OAs

With replicated CPM, an SA orchestration has a *replication degree* k. That is, every activity is assigned with a list of $k+1$ OAs. The first OA in the list, called the *active* OA, is responsible for the interpretation of the message. The rest k OAs are *backup* OAs. For message c, we use $c.A$ for the active OA and $c.\mathcal{A}$ for the backup OAs. We also use $c.\mathcal{A}^+$ for the list of both c's active and backup OAs.

One of our primary goals for the selection of backup OAs is to reuse stored states and keep the run-time overhead of services orchestration as low as possible. The selection is based on the following observations:

- Every OA assigned with some activity for the orchestration will sooner or later obtain some information about the orchestration and this information would overlap with some backup information.
- To keep an OA updated with the information about an OA it backs up, it is often sufficient to send it the deltas of the latest changes, which are typically small fractions of the entire information.
- The amount of overlapping information, and therefore the sizes of the deltas, depends on the freshness of the currently stored information at OAs.

An important property of backup selection is that the backups of an OA can be unambiguously calculated by any OA at any time of the orchestration. This simplifies the handling of events like OA crashes.

The selection algorithm is built on *OA graphs* (OAG) of orchestrations. An OAG is first obtained with a projection of the control flow of the SA to the assigned OAs. If the number of OAs in an OAG is not sufficient for the number of backup candidates, it is extended with more OAs.

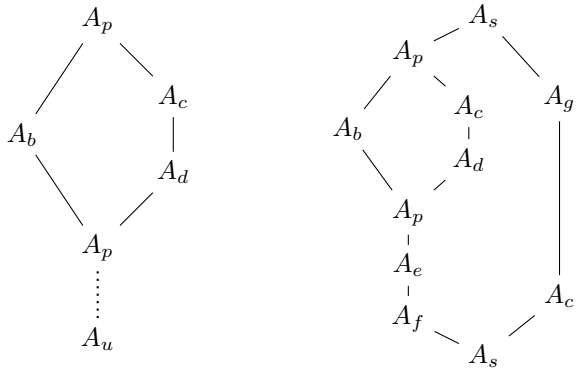
Fig. 6 shows the OAGs of an orchestration of the example SA p of Fig. 1 and an extension s with more OAs. In the OAG of p, A_p is a *parent* of A_b and A_c. If an OA is assigned to consecutive orchestration activities, the OA appears as a single node in the OAG. For example, if both $invoke(S_c, c)$ and $invoke(S_d, d, \bar{d})$ were assigned to A_d, only a single A_d node would have appeared in the OAG. On the other hand, the same OA may appear multiple times in an OAG if it is assigned to activities separated by other OAs. For example, there are two A_p nodes in the OAGs. Parallel branches are ordered. The ordering of branches are decided when an SA is initialized for orchestration. The general rule is that a branch with more orchestration activities has higher priority. For example, the branch with A_b has more orchestration activities than the other branch when n of the loop is larger than 2. In Fig. 6, a branch on the left has higher priority than a branch on the right.

The number of OAs in an OAG is the *degree* of the OAG. It determines the number of backup candidates each OA may have. If an orchestration of p requires that every OA should have 4 backup candidates, the minimum degree of the OAG is 5. This can be obtained by appending one more OA to the youngest node A_p, as A_u in Fig. 6. The selection of A_u is based on the information in the routing component, such as geographic distances.

The backup candidates of an OA A are selected with the following priority order:

S1. OAs of A's enclosing scopes have higher priorities than OAs of lower level nested scopes.
  a) Scopes closer to A have higher priorities.

S2. In a scope, OAs of the same branch have higher priorities than OAs of other branches.
In A's branch,
  a) Ascendant OAs have higher priorities than descendant OAs.



OAG of the example SA p OAG of an extended SA s

Figure 6. OA graph for backup selection.

b) OAs closer to A have higher priorities.

Among the other branches,

c) OAs of a higher-priority branch have higher priorities.

d) In the same branch, OAs closer to the scope manager have higher priorities.

Table III shows the lists of backup candidates (with length 4 for p and 7 for s) of the OAs for the two OAGs in Fig. 6.

TABLE III. BACKUP CANDIDATES

OA	p $(length = 4)$	s $(length = 7)$
A_s		A_p, A_e, A_f, A_g, A_c, A_b, A_d
A_p	A_b, A_c, A_d,A_u	A_s, A_e, A_f, A_b, A_c, A_d, A_g
A_b	A_p, A_c, A_d,A_u	A_p, A_c, A_d, A_s, A_e, A_f, A_g
A_c	A_p, A_d, A_b,A_u	A_p, A_d, A_b, A_s, A_e, A_f, A_g
A_d	A_c, A_p, A_b,A_u	A_c, A_p, A_b, A_s, A_e, A_f, A_g
A_e		A_p, A_s, A_f, A_g, A_c, A_b, A_d
A_f		A_e, A_p, A_s, A_g, A_c, A_b, A_d
A_g		A_s, A_c, A_p, A_e, A_f, A_b, A_d
A_c		A_g, A_s, A_p, A_e, A_f, A_b, A_d

The table only contains A_p and A_s once for each SA. The reason is that the backups for A_p is computed when A_p becomes a scope manager and it stays active until the end of the scope.

As an example, the backups for A_e, are selected according to the following rules of the selection algorithm: A_p (S1.a, S2.a, S2.b), A_s (S1.a, S2.a), A_f (S1.a, S2), A_g (S1.a, S2.c), A_c (S1.a), A_b (S2.c) and A_d.

During an orchestration, $c.\mathcal{A}^+$, the actual active and backup OAs for message c are selected from the first $k+1$ available OAs in the candidates OAs obtained from the OAG.

B. Normal execution

Every CPM message contains an integer k as the replication degree of the current branch, an OAG of degree l ($l > k$) and a list of actual active OA and backups.

In addition, every message has a *timestamp* that can be used to check causal relations between messages. A timestamp is of the form $[b_0, n_0] \cdot [b_1, n_1] \cdot \ldots$, where $b_0, b_1, \ldots$ are the unique identifies of the branches which the message is part of, and $n_0, n_1, \ldots$ are the sequence numbers in the branches. As shown in Fig. 7, in the beginning, there is only one branch (0). After a $fork$, two new branches $(0,0)$ and $(0,1)$ are created. The $orch$ message has sequence number 0 in branch (0). All

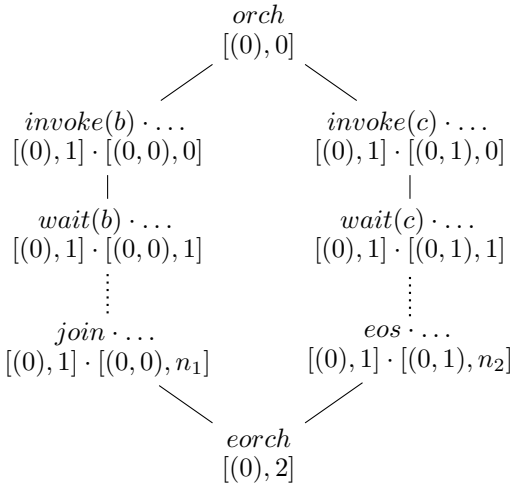


Figure 7. Message timestamps.

messages in the new branches have the same sequence number 1 in the parent branch (0), but different sequence numbers 0, 1, ..., in the new child branches (0.0) and $(0,1)$.

To compare the causality of two messages m^1 and m^2, we first get the longest prefix of their timestamps such that $b_0^1 = b_0^2, \ldots, b_i^1 = b_i^2$ $(i \geq 0)$. Message m^1 *happens before* Message m^2 in the same SA execution, denoted $m^1 \prec m^2$ or $m^2 \succ m^1$, if $n_0^1 = n_0^2, \ldots, n_{i-1}^1 = n_{i-1}^2$ and $n_i^1 < n_i^2$. Messages m^1 and m^2 are *concurrent*, denoted $m^1 \parallel m^2$, if $n_0^1 = n_0^2, \ldots, n_i^1 = n_i^2$.

Suppose that OA A, after interpreting a remote message c_0 and some local interpretations, is currently interpreting message c. Suppose further that the current scope manager and its backups are $c.S$ and $c.\mathcal{S}$ (and $c.\mathcal{S}^+ = \{c.S\} \cup c.\mathcal{S}$). The following are the steps related with sending messages during the orchestration of a normal execution:

C1. When the orchestration of a branch is moving away from A with CPM message c:
   a) Select $c.\mathcal{A}^+$.
   b) Send to $c.\mathcal{A}^+$ message c (or its delta).
   c) Notify $c.\mathcal{S}^+ \cup c_0.\mathcal{A} - c.\mathcal{A}^+$ about the move with message m. m contains two sets of OAs $c.\mathcal{S}^+$ and $\mathcal{G} = c_0.\mathcal{A} - c.\mathcal{A}^+$.

C2. When A stores a local message c in its PM, it also sends the delta of the message c_A to $c_0.\mathcal{A}$.

Step C1.a selects the next active OA and its backups according to the availability of OAs obtained from its RT. Step C1.b extends the destination of a CPM message to include the backups. Step C1.c has two purposes: 1) it extends a scope message to include the scope manager's backups ($c.\mathcal{S}^+$); 2) it informs some of A's backups ($\mathcal{G}$, which no longer backup the subsequent states of the same SA) to purge the backup states. Step C2 informs A's backups about its own state changes.

$c.\mathcal{S}^+$ in step C1.c was selected when the corresponding $scope$ element was interpreted. Step C1.c does not check the availability of the scope manager like step C1.a. The unavailability of an OA that has been active, like a scope manager, is handled in Subsection V-C.

Some messaging overhead is reduced when OAs play multiple roles. For example, when $c.\mathcal{A} = \{A\}$, which is typically true for $k = 1$ (according to the backup selection rules), step

C1.b does not involve any additional remote message than a non-replicated orchestration.

When OA A_r receives a CPM message c (or delta), it does the following:

R1. Ignore c if A_r has already received a message c' such that $c' \succ c$.
R2. If $A_r = c.A$, interpret c.
R3. If $A_r \in c.\mathcal{A}$, store or update backup status of $c.A$.
R4. If $A_r = c.S$, update scope state in SR.
R5. If $A_r \in c.\mathcal{S}$, update BSR.

If a message of a later stage of the same SA execution has already been processed, the newly arrived message is ignored (step R1). The message is handled depending on whether the receiver is an active OA (step R2), a backup OA (step R3), an active scope manager (step R3) or a backup scope manager (step R4).

When OA A_r receives a message m, notifying that an orchestration is moving from A to A', it does the following:

M1. Ignore m if A_r has already received a scope message m' such that $m' \succ m$.
M2. If $A_r = m.S$, update the status of scope in the SR.
M3. If $A_r \in m.\mathcal{S}$, update the backup status of the scope in BSR.
M4. If $A_r \in m.\mathcal{G}$, purge backup status of A.

Notice that in some situations, $c.\mathcal{A}^+ \cap c.\mathcal{S}^+ \neq \emptyset$, the tasks for steps M2 and M3 are done in R4 and R5. In general, the more these sets overlap, the more overhead is avoided.

C. Handling unavailability of OAs

When an OA becomes unavailable, its tasks for services orchestration, either as an active or backup OA, are taken over by other OAs. There are two types of tasks: interpretation of CPM messages and management of scopes. In this subsection, we focus on the first type, i.e., to continue interpreting CPM messages when an OA becomes unavailable. The steps to continue scope management is almost the same.

The unavailability of OAs is handled on a per-message basis, or a per-branch basis, because every CPM message represents an SA branch. When an OA in $c.\mathcal{A}^+$ becomes unavailable, it is always the highest ranked available OA in $c.\mathcal{A}^+$ to take the responsibility of handling the unavailability.

An OA becomes unavailable either when it leaves the OA network on purpose, or when it crashes or is disconnected due to some network failure. Before OA A leaves on purpose, it notifies the highest ranked available OA in $c.\mathcal{A}^+ - \{A\}$ for every message c in its PM and BM about its leaving. An OA A_r does the following when receiving this message:

L1. If A is the highest ranked OA in $c.\mathcal{A}^+$, A_r takes over as the actual active OA of c.
L2. Add a new OA to $c.\mathcal{A}^+$ according to the OAG and inform the new $c.\mathcal{A}^+$ about the latest update of c.

When an OA crashes or is disconnected from the network, its unavailability is detected when another OA is unable to send it a message. Because the OAs exchange routing messages regularly [6], the unavailability is detected in short time. Generally, the busier the OA network, the shorter the detection time. As soon as an unavailability is detected, it is propagated to the entire OA network.

When an OA A_r is notified of the unavailability of A, it finds relevant CPM messages in its PM and BM. A message c is relevant if $A \in c.\mathcal{A}^+$. For each such message c, it does the following:

U1. If A_r is the highest ranked available OA in $c.\mathcal{A}^+ - \{A\}$, do L1 and L2.

With respect to correctness, think of a message as representing a particular step of a branch. Because only the highest ranked available backup OA takes over the role as the new active OA of a message when the current active OA becomes unavailable (and once an OA is detected as unavailable, it will not be re-assigned to the same process execution when it becomes available again), it is impossible for two OAs to simultaneously take over as the new active OA of the same message.

However, backups of different messages of the same branch may coexist in different OAs. Consequently, different OAs may independently take over the role as the active OAs of different steps of the same branch. This does no harm when business critical services enforce the at-most-once execution model. In addition, if a scope manager observes that two OAs are responsible for the orchestration of the same branch, it kills the activities represented by the outdated messages. Eventually, the active OA of the most up-to-date message wins as the only active OA of the branch.

To make the last point clearer, consider this particular situation: OA A becomes unavailable just after an orchestration moved to the next OA A', and the notification of the unavailability arrives to A_r before the notification of the move (steps C1.b and C1.c in Subsection V-B). In this situation, A_r may take over and repeat the work that A had just finished before it became unavailable. The repeated work will eventually arrive at A'. By checking the timestamp of the message (step R1 in Subsection V-B), A' can figure out that the orchestration of this branch has already passed over this stage. The same is also detected by the scope manager (step M1). In the worst case, if a service invocation is repeated, a business-critical service will return with an exception due to the at-most-once semantics.

The last issue will not occur for replicated scope managers, because a scope manager never moves from OA to OA in the basic CPM scheme.

At this point, it should be clear that the replication scheme can tolerate up to k crashes during the time interval between the detection and the handling of an unavailability.

VI. PERFORMANCE

We developed an OA prototype that runs in a simulator [9] for performance study. We study the performance of OAs with different degrees of replication and at different workload.

In our experiment, there are 100 SPs, 10 of the which are OAs as well. That is, these 10 sites both process service invocations and contribute to orchestration of services. Every OA covers 10 SPs. The distances between an OA and the SPs it covers are relatively short. An SP spends on average 100ms to process a service invocation. An OA spends on average 10ms to interpret a CPM message, and 1ms to handle a scope message, backup message or purge message. We model the workload with *multiprogramming levels* (MPLs) of SPs, which is the number of concurrent service operations it executes most of the time. Initially, a fixed number of SA executions are fed into the system. A new SA execution starts as soon as an existing one terminates. An SA execution consists of 4 operation invocations to different randomly chosen SPs.

 ISBN: 978-1-61208-337-7

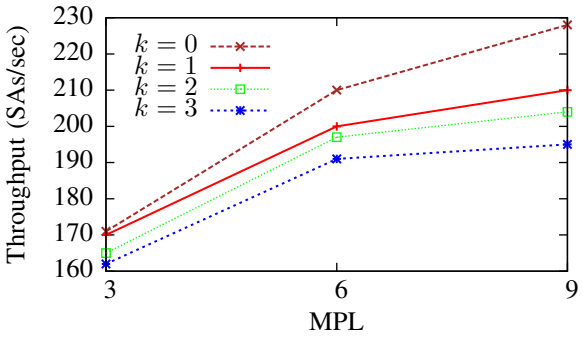


Figure 8. Throughput of 100 SPs.

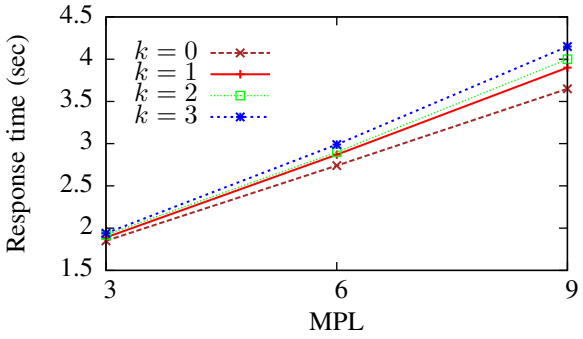


Figure 9. Response time of SAs.

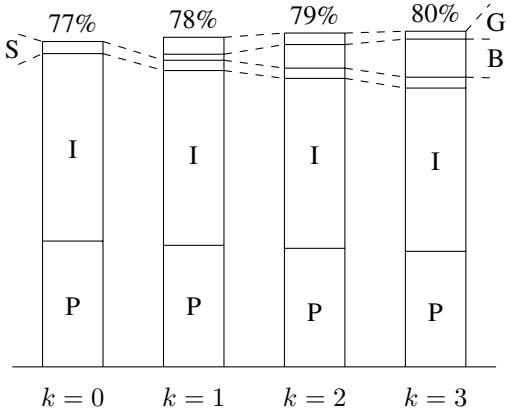
Fig. 8 shows the aggregate throughput of all SPs (measured in the number of completed SA executions per second). Fig. 9 shows the average response time of the SAs.

Fig. 10 shows the average resource utilization at OAs when the SP MPL is 6. We only show the resource utilization at one MPL, because although the total resource utilization varies at different MPLs, the proportion of different kinds of message handling is almost the same through all MPLs. As the degree of replication increases, the overhead of backup management ("B" and "G") increases, and the capability of normal service orchestration ("I" and "S") and service operation execution ("P") decreases. Consequently, the overall SP throughput decreases and SA response time increases, as shown in Fig. 8 and Fig. 9.

It is interesting to notice that when $k = 1$, the overhead of backing up orchestration states ("B") is less than the overhead of purging the backup states ("G"). The reason for this is that when an OA A forwards the orchestration to the next OA A', $c.\mathcal{A}^+ = \{A, A'\}$ in step C1 of Subsection V-B. In other words, A already has the state locally and the overhead of backing up the state is therefore low.

It is also interesting to notice that when k increases, the overhead of purging the backup states ("G") decreases. This is because an OA backups up the states of several stages of the same orchestration. When it store the backup state of a new stage, it also purges the state of an earlier stage. In other words, the larger overlap of $c_0.\mathcal{A}$ and $c.\mathcal{A}^+$ in step C1.c of Subsection V-B leads to the decrease of "G".

When an OA becomes unavailable, other OAs will handle the unavailability. We expected that this will cause a sudden increase of workload which will influence the overall performance of the system. For example, when k is 2 and SP MPL is 6, an OA covering 10 SPs is handling (most of the



P: service process, I: message interpretation, S: scope management
B: store/update of backup states, G: purge of backup states

Figure 10. Resource utilization at OAs at MPL 6.

time) 60 CPM messages and backing up 120 for other OAs. If an OA crashes, 180 messages will be handled by other OAs. However, in our experiments, we could not observe significant overall performance hiccup. The main observable difference in overall performance is that MPL of OAs has increased nearly 10%, both during handling of the unavailability and afterwards. It turns out that the messages that the unavailable OA was actively orchestrating (60 in this example) were the primary contributor to the increase of load at other OAs. The backup messages (120 in this example) contributed only very little to the increase of load at other OAs. More precisely, it is primarily the "I" part in Fig. 10 that contributed to the increase of load at the remaining OAs.

VII. Related Work

Decentralized orchestration in SOC research can be categorized into instantiation-based or messaging-based [5]. An instantiation-based approach [10][11][12][13][14] instantiates in advance a composition with resource and control allocation in the distributed environment. The allocated resources and control are responsible for the orchestration of the subsequent executions of the same composition. This approach is therefore more suitable for enterprise applications where allocation of resources is possible, and compositions are stable and are typically repeated many times [5]. In messaging-based approaches [5][10][15][16], information like execution order of activities is carried in messages. Since no resource or control is allocated in the distributed environment before an execution starts, messaging-based approaches would be more appropriate for orchestration where either the compositions or the environment are so dynamic that pre-allocation of resources is impractical.

The focus of research on reliable services orchestration has been on dealing with failures of constituent services, mostly based on compensation-based recovery [5][12][13][16]. Little work is done on dealing with failures of orchestration engines or agents.

Several replication schemes have been proposed in the research area of data streams and continuous queries. Gedik and Liu [17] applied a passive or backup replication mechanism to executions of continuous queries. A continuous query is executed on peers with matching ids. The selection of replicas

 ISBN: 978-1-61208-337-7

or backups is based on peer ids and neighbor proximity of the peer-to-peer network. In [18], data flow from sensors to data processing programs through an overlay network of peers. Peers are grouped into cells. Active replication is applied to the peers in the same cells to enhance the availability of the data flows. Martin, Fetzer and Brito [19] proposed an active replication scheme to a stream variant of map-reduce system consisting of stages of map-reduce operators. Replication is applied among data partitions of the same stage. The focus is on utilizing unused CPU cycles for replication. Zhang et al. [20] introduced a hybrid active/passive replication scheme to a peer-to-peer stream processing system to deal with transient failures due to high workload. It dynamically switches between active and passive schemes according to the workload in order to utilize the best part of both schemes.

The key difference of the afore-mentioned replication work and ours is that in continuous queries or data stream processing, tasks assigned to processing agents or peers are long lasting. It is therefore more suitable to have a relatively stable set of replicas and even special-purpose multicast communications among them.

Continuation-passing messaging was presented in more detail in our early work [5]. This early approach was however too intrusive. It requires that service providers support message interpretation. Although this might be arguably acceptable for enterprise applications, it is too strong a requirement for open services. Organization of OA networks for orchestration was later presented in [6]. Support for exception handling and rollback due to service failures was also presented in more detail in [5].

VIII. Conclusion

We presented a replication scheme for decentralized orchestration of open services with continuation-passing messaging. The scheme utilizes the knowledge about the control structure that is encapsulated in messages and the run-time state that is already spread in the distributed environment to enhance the availability of the orchestration. For a degree-k replicated orchestration, every branch can tolerate simultaneous crashes of up to k orchestration agents. Our performance study shows the overhead of replication during normal services orchestration.

There are still a number of issues to be addressed before the new approach can be practically adopted.

Security is always an important concern of distributed applications. We have not worked on security issues yet, but our approach is already useful when used in special cases. For example, if the orchestration agents are deployed at geographically different places by the same organization or a set of trusted applications, these agents can be used as a smart pool of orchestration engines where the orchestration activities are dispatched to the most appropriate engines.

The performance study shows that replication does incur a performance penalty. An incentive model would encourage more service providers to offer as orchestration agents. For example, applications that offer orchestration capabilities should have higher priority when scheduled and should be more entitled to higher degree of replication.

References

[1] C. Peltz, "Web services orchestration and choreography," Computer, vol. 36, no. 10, 2003, pp. 46–52.

[2] Amazon SWF: The Amazon Simple Workflow Service, [retrieved: April, 2014] http://aws.amazon.com/documentation/swf/.

[3] M. Wieland, K. Görlach, D. Schumm, and F. Leymann, "Towards reference passing in web service and workflow-based applications," in Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference (EDOC), Auckland, New Zealand, 2009, pp. 109–118.

[4] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "PeerSoN: P2P social networking: early experiences and insights," in Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (SNS), Nuremberg, Germany, 2009, pp. 46–52.

[5] W. Yu and A. A. M. M. Haque, "Decentralised web-services orchestration with continuation-passing messaging," IJWGS, vol. 7, no. 3, 2011, pp. 304–330.

[6] A. A. M. M. Haque, W. Yu, A. Andersen, and R. Karlsen, "Peer-to-peer orchestration of web mashups," in The 14th International Conference on Information Integration and Web-based Applications and Services (iiWAS), Bali, Indonesia. ACM, 2012, pp. 294–298.

[7] Web Services Business Process Execution Language (WS-BPEL) Version 2.0, Organization for the Advancement of Structured Information Standards (OASIS), April 2007, [retrieved: April, 2014] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[8] SERF, [retrieved: April, 2014] http://www.serfdom.io/.

[9] A. Varga, "OMNeT++," in Modeling and Tools for Network Simulation, K. Wehrle, M. Günes, and J. Gross, Eds. Springer, 2010, pp. 35–59.

[10] A. Barker and R. Buyya, "Decentralised orchestration of service-oriented scientific workflows," in Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER), Noordwijkerhout, Netherlands, 2011, pp. 222–231.

[11] B. Benatallah, M. Dumas, and Q. Z. Sheng, "Facilitating the rapid development and scalable orchestration of composite web services," Distributed and Parallel Databases, vol. 17, no. 1, 2005, pp. 5–37.

[12] G. Chafle, S. Chandra, V. Mann, and M. G. Nanda, "Decentralized orchestration of composite web services," in Proceedings of the 13th international conference on World Wide Web (WWW), New York, USA, 2004, pp. 134–143.

[13] G. J. Fakas and B. Karakostas, "A peer to peer (P2P) architecture for dynamic workflow management," Information & SW Technology, vol. 46, no. 6, 2004, pp. 423–431.

[14] P. Muth, D. Wodtke, J. Weißenfels, A. K. Dittrich, and G. Weikum, "From centralized workflow specification to distributed workflow execution," J. Intell. Inf. Syst., vol. 10, no. 2, 1998, pp. 159–184.

[15] D. Martin, D. Wutke, and F. Leymann, "A novel approach to decentralized workflow enactment," in Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC), Munich, Germany, 2008, pp. 127–136.

[16] T. Möller and H. Schuldt, "A platform to support decentralized and dynamically distributed P2P composite OWL-S service execution," in Proceedings of the 2nd Workshop on Middleware for Service Oriented Computing (MW4SOC), Newport Beach, California, USA, 2007, pp. 24–29.

[17] B. Gedik and L. Liu, "A scalable peer-to-peer architecture for distributed information monitoring applications," IEEE Transactions on Computers, vol. 54, no. 6, 2005, pp. 767–782.

[18] R. Martins, P. Narasimhan, L. Lopes, and F. Silva, "Lightweight fault-tolerance for peer-to-peer middleware," in Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems (SRDS), New Delhi, India, 2010, pp. 313–317.

[19] A. Martin, C. Fetzer, and A. Brito, "Active replication at (almost) no cost," in Proceedings of the 30th IEEE Symposium on Reliable Distributed Systems (SRDS), Madrid, Spain, 2011, pp. 21–30.

[20] Z. Zhang et al., "A hybrid approach to high availability in stream processing systems," in Proceedings of IEEE 30th International Conference on Distributed Computing Systems (ICDCS), Genova, Italy, 2010, pp. 138–148.

 ISBN: 978-1-61208-337-7

Why are Reputation Systems Absent from Cloud Services: Reason and Solution

Lianyong Qi, Jiancheng Ni, Chao Yan, Xiaona Xia, Chunmei Ma
Computer Science College
Qufu Normal University
Rizhao, 276826, China
Email: {lianyongqi@gmail.com, nijch@163.com, yanchao@qfnu.edu.cn, xiaxn@sina.com, rsmcm@163.com}

***Abstract*—Feedback rating-based reputation system is usually considered as an effective approach to build the trust between cloud users and cloud providers. However, unfortunately, such a reputation system is absent from the present major cloud providers, e.g., *Amazon*, *Google* and *Microsoft*, which embarrasses a cloud user from selecting a trusted cloud service from a cloud provider. In view of this challenge, in this paper, we first analyze the cloud characteristics, and study why reputation systems are absent from cloud providers, from perspectives of cloud provider and cloud user respectively. Afterwards, two reputation systems of popular e-Commerce service platforms, i.e., *Amazon.com* and *eBay.com*, are investigated respectively. Finally, a reputation system tailored to cloud services, i.e., Cloud Reputation System (CRS) is brought forth. CRS not only considers the advantages of e-Commerce reputation systems, but also adapts to the cloud characteristics. We believe that the proposed CRS is helpful, for building the trust between cloud users and cloud providers in the future.**

Keywords-Cloud user*; *Cloud provider*; *Trust*; *Feedback rating*; *Reputation system*; *Service quality

I. INTRODUCTION

As a natural evolution of Services Computing, Cloud Computing has recently gained more and more attentions, from both academic and industry domains [1][2]. By delivering various computing resources in a pay-as-you-go manner, Cloud Computing is helping human to realize the long-held dream of computing-as-a-utility. In the cloud environment, a cloud provider could share its idle computing resources for additional income. While on the other hand, a cloud user can also benefit from moving his/her business applications towards cloud, so as to enjoy an easy-to-deploy, maintenance-free and cost-effective business competitive advantage [3].

However, due to the open and dynamic nature of cloud environment, the Quality of Service (QoS) of a cloud service is not always as good as advertised; even a Service Level Agreement (SLA) contract is made beforehand between a cloud user and a cloud provider [4]. We analyzed the reasons as follows. First, inside the cloud provider, we cannot expect the availability of a cloud service is always 100% in a billing cycle (e.g., numerous reported outage incidents [5]). Besides, inside attacks and damages are also possible (for example, *Google* has to fire the employees for their illegitimate operation on user data [6]). Second, outside the cloud provider, we cannot precisely predict the actual execution context (e.g., *network delay*), when a cloud user requests a cloud service from a cloud provider. Besides, malicious attacks from competitors are also inevitable in cloud environment [2]. Therefore, the delivered service quality of a cloud provider is fluctuant, and sometimes may not meet the quality expectation of the cloud user. In other words, cloud provider (or the service delivered by a cloud provider) is not always 'trusted' as promised. Therefore, it is of great significance to build trust between cloud users and cloud providers.

Feedback rating and review are regarded as an effective manner to build trust between service providers and service users, and now widely adopted in present popular e-Commerce service platforms [7]. For example, if one buys a smartphone from an e-Commerce platform, he/she can leave a negative or positive rating (1-star to 5-stars, and 5-star is the best) or review, according to his/her satisfaction towards the smartphone quality and shipping service. However, compared with e-Commerce, present major cloud providers, e.g., *Amazon*, *Google* and *Microsoft* lack such a reputation system. In this situation, if one requests a cloud service from a cloud provider, e.g., *Amazon*, he/she has no way to evaluate and predict the cloud service quality before the service is delivered and executed. Therefore, the absence of reputation system makes it a challenge for cloud users to select a trusted cloud service from cloud providers.

In view of this challenge, in this paper, we study the reasons that reputation systems are absent from cloud providers, and put forward a reputation system tailored to cloud service delivery. The remainder of this paper is organized as below. In Section 2, we analyze the cloud characteristics and study why reputation systems are absent from the present cloud providers. Afterwards, in Section 3, two reputation systems of e-Commerce (i.e., *Amazon.com* and *eBay.com*) are investigated respectively. In Section 4, a reputation system for cloud services, i.e., CRS (Cloud Reputation System, CRS) is put forward, by considering the cloud characteristics analyzed in Section 2 and the e-Commerce reputation systems investigated in Section 3. Related work and comparison analyses are introduced in Section 5, and finally, conclusions are drawn in Section 6.

 ISBN: 978-1-61208-337-7

II. ABSENCE OF REPUTATION SYSTEM FROM CLOUD PROVIDERS: THE REASONS

Rating-based reputation system is a good supplement for calculating the trustworthiness of a cloud service before its delivery. However, as far as we know, the present major cloud providers (e.g., *Amazon*, *IBM* and *Microsoft*) do not support such a reputation system. In this section, we analyze the reasons from the perspectives of cloud provider and cloud user, which are listed briefly in Table 1.

TABLE I. ABSENCE REASONS OF REPUTATION SYSTEM FROM CLOUD

Perspective	Id	Reason
Cloud provider	**1**	Lack of incentive
	2	Have confidence in delivering high service quality
	3	Fear for malicious ratings
Cloud user	**1**	Hard to rate a cloud service with a long running period
	2	Hard to rate a cloud service in service combination
	3	Hard to observe the necessary QoS data for rating

A. Reasons From the Perspective of Cloud Provider

In this subsection, we study the reasons that reputation systems are absent from cloud, from the perspective of cloud provider.

(1) Lack of incentive

At present, the big and competitive companies constitute the majority of cloud providers. For example, *Amazon* occupies 80%-90% market share of IaaS [8]. In this situation, a cloud user has few choices when he/she requests a cloud service; hence, the big cloud provider, e.g., *Amazon* lacks incentive to build its reputation system. Besides, no competition exists inside a cloud provider. For example, if a cloud user requests elastic computing resources from *Amazon*, he/she has no other choice but to select *EC2* service, because only *EC2* service is able to provide the elastic computing functionality inside *Amazon*. In this situation, *EC2* faces no competition inside *Amazon*. Therefore, from the perspective of *Amazon*, it is regarded as unnecessary to measure and publish the reputation of *EC2*, even if different cloud users may experience different service quality from *EC2*.

(2) Have confidence in delivering high service quality

The big cloud providers, such as *Amazon*, deliver rich cloud services and have advanced techniques to ensure that a high quality service is provided. Therefore, the big cloud providers often have confidence in their delivered service quality, and regard it unnecessary to build a reputation system for their cloud services. For example, as Fig.1 shows, *Amazon* declares 99.99% service availability in its SaaS SLA contract, and different compensation rates are available if the agreed availability is violated [3].

However, as analyzed in Section 1, the service quality delivered by cloud providers is not always as high as promised, due to the malicious attacks from outside, or dynamic change of network environment. Besides, the simple compensation mechanism is not suitable for all cloud users, when SLA agreement is violated. For example, if a critical task is failed due to the poor quality of a cloud service, the user may prefer to leave a lowest rating (e.g., 1-star) to the cloud service, rather than receive a compensation of $100.

Service Level Agreement

Availability

- 99.99% uptime

Compensation

- Percentage of total charges paid by cloud user

UPTIME (PER 15 MIN)	COMPENSATION
99.99% - 100%	0%
98.00% - 99.98%	5%
97.00% - 97.99%	10%
95.00% - 96.99%	20%
< 95.00%	50%

Figure1. An example of SLA contract

(3) Fear for malicious ratings

After a user invoked a service, he/she can give the service a feedback rating, based on the perceived service quality and his/her quality preference. Therefore, the feedback rating is rather subjective, and the feedback rating-based reputation system is vulnerable to the malicious attacks. For example, a malicious user may give a 1-star rating to a 5-star delivered service, or give a 5-star rating to a 1-star delivered service, for commerce or competition reasons. Similar fears are also existent for the cloud providers, because a good reputation accumulated within a long period could be easily damaged by a malicious user rating. Therefore, from the perspective of cloud provider, it prefers to leave the reputation system empty, rather than have its service reputation attacked by potential malicious cloud users.

B. Reasons From the Perspective of Cloud user

Different from the traditional web service, cloud services have some particular characteristics. Next, we will introduce these characteristics, and analyze the reasons that reputation systems are absent from cloud, from the perspective of cloud provider.

(1) Hard to rate a cloud service with a long running period

Different from the traditional web services whose running period is short, the running period of a cloud service is usually long, e.g., one year, during which the cloud provider will deliver its cloud services continuously. In this situation, it is hard for a cloud user to rate a cloud service during its long running period. First, a cloud user

 ISBN: 978-1-61208-337-7

cannot wait to give his/her final rating until the cloud service's delivery ends, because the waiting time is too long (e.g., a cloud user has to wait for one year, in order to rate his/her requested one-year-period *Email service* from *Google*). Second, the service quality of a cloud service may change constantly, during the service's long running period. Hence, a cloud user cannot give a fair and accurate rating, towards the dynamically changed service quality of a cloud service.

(2) Hard to rate a cloud service in service combination

Generally, a cloud provider delivers its cloud services in the form of service combination. For example, Table 2 lists four cloud service combinations advertised by *Amazon EC2* [9], i.e., {Small instance, Middle instance, Large instance and Extra-large instance}, where each instance is a combination of four categories of cloud services {Memory, EC2 Computing Unit, Local Storage, Platform}.

TABLE II. AN INSTANCE OF CLOUD SERVICE COMBINATION

	Memory (GiB)	EC2 Computing Unit	Local Storage (GB)	Platform (bit)
Small	1.7	1	160	32 or 64
Middle	3.75	2	410	32 or 64
Large	7.5	4	850	64
Extra-large	15	8	1690	64

In this situation, a cloud user can only give an global rating towards the whole service combination instance. For example, a cloud user gives a '4-star' rating to service combination 'Middle instance' in Table 2. Obviously, this rating is a global rating towards the quality performance of combination (e.g., 'Middle instance'), not a local rating for a single cloud service (e.g., '410 GB Local storage' in 'Middle instance'). In this situation, the global rating has little effect in evaluating the service quality of a single cloud service; even if a global rating is given by a cloud user. For example, if a cloud user gives a lowest '1-star' rating to 'Middle instance', we cannot determine whether the bad rating is caused by the poor quality of '3.75 GiB Memory' or '2 EC2 computing unit' or '410 GB Local Storage' or '32 or 64 Platform'.

(3) Hard to observe the necessary QoS data for rating

In cloud environment, business applications of users are deployed and executed on the remote servers of cloud providers, not locally. Therefore, a cloud user has little control on its business execution, and thereby cannot observe the detailed QoS data associated with cloud service delivery, e.g., the actually delivered *disk I/O*, *response time of storage service*. Although several toolkits have been developed to monitor the QoS data of cloud service delivery, e.g., *Amazon CloudWatch* [10], the monitoring range is limited and the monitoring accuracy is doubtable. For example, if a cloud user utilizes *CloudWatch* to monitor *EC2* service, the authenticity of monitored QoS data is doubtable, as both *CloudWatch* and *EC2* are developed by *Amazon*. Therefore, it is hard for a cloud user to rate a cloud service, based on the little observed QoS data.

Based on the above reason analyses, we have identified the obstacles that lead to the absence of reputation system from cloud, from perspectives of cloud provider and cloud user. Next, two reputation systems in e-Commerce, e.g., on-line *Amazon.com* and *eBay.com* will be investigated respectively, which could be regarded as beneficial references for building a reputation system for cloud services in the future.

III. INVESTIGATION OF REPUTATION SYSTEMS IN E-COMMERCE

Although few cloud providers also build their reputation systems, e.g., *Rackspace Inc.* [11], the reputation system is rather simple and cannot accommodate the cloud service delivery very well. In this section, the reputation systems of on-line *Amazon.com* and *eBay.com* will be investigated respectively, which are beneficial references for building a reputation system for cloud services, as e-Commerce and cloud provider both deliver their 'services' to the public.

A. Reputation system of Amazon.com

As a successful on-line mall that delivers thousands of products to people all over the world, *Amazon.com* [12] is famous for its delivered high-quality products and objective reputation system. For each product in *Amazon.com*, a reputation is built, which mainly consists of the following two components: **user rating** and **user review**.

(1) User rating

For each product, a user can leave a feedback rating from '1-star' to '5-star' ('5-star' is the best), to indicate his/her satisfaction degree towards the product quality or service quality. Then according to the ratings from all users, an average rating is assigned to a product. For example, for 'Kindle Fire HD' product, totally 824 users give their ratings, where there are 34 '1-star' ratings, 37 '2-star' ratings, 115 '3-star' ratings, 232 '4-star' ratings and 406 '5-star' ratings. Therefore, the average rating for 'Kindle Fire HD' is '4.1-star'. This average rating could reflect the user-perceived product quality approximately. Besides, the user rating is not fixed, but variable. For example, if 'Kindle Fire HD' cannot work after one month use, the user may revise the pre-assigned '5-star' rating to '1-star' rating, so as to express his/her extreme anger. This kind of variable user rating is really suitable for rating the quality of long-lifecycle products.

 ISBN: 978-1-61208-337-7

(2) User review

Besides user rating, *Amazon.com* allows users to give their reviews about a product. Considering the above example, 824 reviews are available for 'Kindle Fire HD' product. In a review, a user could describe his/her satisfaction or dissatisfaction, as well as the reasons. Moreover, user B can rate a review from user A ('helpful' or 'not helpful'), which can reflect whether user A's review is helpful to user B. For example, if 100 users read a review from user A, and 98 users consider the review helpful, then a ratio '98/100' is assigned to the review from user A. In this way, *Amazon.com* can avoid some malicious user ratings and reviews.

Despite of the above advantages, the reputation system of *Amazon.com* still has some limitations. For example, anyone can rate a product; even if he/she did not buy this product from *Amazon.com*. In this situation, the reputation system could be easily attacked by malicious users, e.g., by *Sybil attacks* [3].

B. Reputation system of eBay.com

Compared with *Amazon.com*, the reputation system of *eBay.com* [13] is more complicated, which mainly consists of three components: **mutual rating**, **user review** and **Quantity sold**.

(1) Mutual rating

Different from *Amazon.com*, the ratings of *eBay.com* are mutual: **buyer rating** and **seller rating**.

Buyer rating: A buyer can rate the service quality of a seller by buyer rating. If buyer A buys a product from seller B, A can give an overall rating to B, i.e., 'Positive' or 'Neutral' or 'Negative'. Moreover, more detailed ratings could be given, according to the four criteria {*Item as described*, *Communication*, *Shipping time*, *Shipping and handling charges*} of B, each of which could be rated from 1-star to 5-star by A. Therefore, A can give one overall rating and four detailed ratings towards B. For example, buyer A's overall rating towards B is 'Positive', and detailed ratings are respectively {5-star, 4-star, 5-star, 4-star} corresponding to the above four criteria. Besides, according to overall ratings from all buyers, seller B is assigned an overall '99.95% Positive' rating (excluding the repeated ratings from the same buyer in one week) by *eBay.com*. Likewise, according to detailed ratings from all buyers, seller B is assigned a detailed {4.9-star, 4.7-star, 5-star, 4.8-star} rating by *eBay.com*.

Seller rating: A seller can also rate the hehavior of a buyer by seller rating. After the buyer rates the seller, the seller can also rate the buyer as 'Positive' or 'Neutral' or 'Negative'. According to the seller ratings from all sellers, a buyer (e.g., A) is assigned an overall seller rating, e.g., '98.5% Positive' by *eBay.com*.

Time factor is also considered in *eBay.com*. For example, both the buyer rating and seller rating should be given in 60 days since a deal is agreed, and could be revised only once in 10 days since the rating is given. Of course, a buyer can also view the past buyer ratings of a seller, e.g., buyer ratings in recent one month, in recent six months or in recent one year.

(2) User review

The user review of *eBay.com* is similar with that of *Amazon.com*, so it will not be discussed repeatedly. The minor difference between them is that: in *eBay.com*, a review should be given in 60 days since a purchase behavior occurs, and can only be revised once in 10 days after its birth.

(3) Quantity sold

For each product, a 'quantity sold' number is assigned by *eBay.com*, to indicate the popularity of the product in a recent period. For example, '1000 sold last month' shows a great confidence of buyers towards a product recently. Although 'quantity sold' is not a direct component of reputation in e-Commerce, it is still regarded as an important factor when evaluating the reputation of a product.

Next, based on the above analyses, we compare the reputation systems of *Amazon.com* and *eBay.com*, from different angles. The comparison results are listed in Table 3, where better reputation strategies are stressed with a darker background color. Here, for some reputation strategies employed, we cannot determine whether it is good or not, such as the last criterion in Table 3, i.e., 'quantity sold'. According to *eBay.com*, 'quantity sold' is a good indicator towards the popularity and quality of a product; however, according to *Amazon.com*, 'quantity sold' is bad because it distracts users' attention from focusing on the product quality itself. As in Table 3, neither of the two reputation systems can outperform the other.

TABLE III. REPUTATION SYSTEM COMPARISONS: AMAZON.COM VS EBAY.COM

Feedback type \ e-Commerce		Amazon.com	eBay.com
User rating	Overall rating	Yes /1-star to 5-star	Yes /Positive or Neutral or Negative
	Detailed rating	No	Yes
	Mutual rating	No	Yes
	Non-user rating	Yes	No
	Revisable	Yes/anytime	Yes/once in 10 days
	Timely rating	Yes/not must	Yes/in 60 days
	Repeated rating	Yes	No
	Malicious rating	Yes/easy	Yes/difficult
	Mandatory rating	No	No
User review	Timely review	Yes/not must	Yes/in 60 days
	Revisable	Yes/anytime	Yes/once in 10 days
	Rating for review	Yes	Yes
	Malicious review	Yes/easy	Yes/difficult
Quantity sold		No	Yes

IV. A REPUTATION SYSTEM FOR CLOUD SERVICES

In this section, a reputation system for cloud services, i.e., CRS is put forward. Here, CRS does not discuss the concrete reputation calculation process of cloud services, instead, CRS provides detailed solutions for solving the obstacles and difficulties introduced in Section 2, when building a reputation system for cloud services. The details of CRS are listed in Table 4. Next, we explain why the proposed solutions can solve the present obstacles.

TABLE IV. CLOUD REPUTATION SYSTEM CRS: OBSTACLE & SOLUTION

ID	Obstacle	Solution
1	Lack of incentive	Number of invocations
2	Have confidence in delivering high service quality	Overall rating
		Detailed rating
3	Fear for malicious ratings	Mutual rating
		NO Non-user rating
		NO Non-user review
		NO repeated rating
4	Hard to rate a cloud service with a long running period	Revisable rating
		Revisable review
		Timely rating(alternative)
		Timely review(alternative)
		Period rating
5	Hard to rate a cloud service in service combination	Detailed rating
6	Hard to observe the necessary QoS data for rating	Detailed rating
		NO Mandatory rating

Obstacle1: Lack of incentive. According to the Bandwagon Effect [14], the more frequently a cloud service is invoked, the more attractive it is for cloud users. Therefore, 'Number of invocations' is of positive significance, for promoting cloud providers to build their respective reputation systems.

Obstacle2: Have confidence in delivering high service quality. The low 'Overall rating' and 'Detailed rating' data can reminder the cloud providers to improve their poor service quality, so as to avoid cloud providers' overconfidence in their delivered service quality.

Obstacle3: Fear for malicious ratings. The proposed 'Mutual rating' can increase the risk of a cloud user, if he/she gives a malicious rating. Besides, the Non-users are not allowed to give a rating or review, by which we can reduce the malicious ratings or reviews from the Non-users. Furthermore, repeated ratings are not allowed, which can increase the cost of a cloud user when he/she gives a malicious rating.

Obstacle4: Hard to rate a cloud service with a long running period. For the cloud services with a long running period, a cloud user can give his/her timely rating or review after he/she invokes the service. Besides, a user can report the latest service quality rating every other period, which is called 'Period rating'. In order to cope with the dynamic changes of service quality during the long running period, users' rating or review towards a cloud service are revisable.

Obstacle5: Hard to rate a cloud service in service combination. Actually, it is difficult to rate a single cloud service in service combination. However, we can make some attempt under some reasonable assumptions. For example, as Table 2 shows, we assume that *Response time* is affected greatly by the single service 'EC2 computing unit' and likewise, *Throughput* is affected greatly by the single service 'Memory'. Under these two assumptions, we can rate single services 'EC2 computing unit' and 'Memory' approximately, through the 'Detailed rating' towards QoS criteria *Response time* and *Throughput*.

Obstacle6: Hard to observe the necessary QoS data for rating. With the limited QoS data that is observed, a cloud user can give its detailed ratings towards few or partial QoS criteria, which is still of positive significance for future cloud service selection. Besides, mandatory rating is not allowed so as to ensure the authenticity.

V. RELATED WORK AND COMPARISON ANALYSES

Cloud computing has exhibited its great advantages in delivering use-on-demand and pay-per-use computing services [1][2][15]. More and more users are moving their business or personal applications towards cloud. However, due to the dynamic and open nature of cloud environment, a cloud service may not deliver a satisfactory quality level as promised in its SLA contract. In other words, from perspective of a cloud user, a cloud service is not always 'trusted' during its delivery period. Many researchers have observed and studied this trust problem.

Academic area. SLA is considered as a feasible manner to build trust between a cloud user and a cloud provider [2]. A cloud provider is regarded as trusted, if its service is delivered with SLA-agreed quality. Sheikh Mahbub Habib, et al. [15] introduces a set of attributes, e.g., *security*, *performance* and *compliance*, to monitor and measure the SLA violation. However, some quality performance declared in SLA is hard to monitor directly. Therefore, as an indirect manner, Monoj Kumar Muchahari, et al. [16] proposes a feedback rating-based trust calculation method, i.e., *TrustCalculator*, to estimate the future quality of a cloud service, based on its past feedback ratings from cloud users. However, the assumed user rating is of a rather simple form, i.e., from 0 to 5, which cannot accommodate the complicated cloud service delivery very well. As malicious rating is possible, S. Wang, et al. [17] proposes a detection method of malicious rating, by comparing the monitored service quality and the expected service quality in SLA. The object of this proposal is to ensure that all the user ratings are real and trusted, not malicious, which has the same function as our proposed 'Mutual rating' and 'Rating for review' in *R3*. Talal H. Noor, et al. [18] proposes a *Trust Feedback Collector* to collect user feedbacks from cloud service delivery. This collector provides an essential foundation, for building our proposed *R3* reputation system in cloud.

 ISBN: 978-1-61208-337-7

Industry area. Compared with the enthusiasm in academic area, few progresses in industry area could be found in building a cloud reputation system. Concretely, only few cloud providers offer their reputation systems, e.g., *Rackspace Inc.* [11]. And the effect is not as good as expected, for example, only 45 user reviews are left in *Rackspace* reputation systems since 2009. In contrast, most major cloud providers, e.g., Amazon, Google and Microsoft don't offer sufficient reputation systems to support the trust evaluation of their cloud services, which is the major motivation of our paper. In view of the disappointed industry status, we analyze the reasons that reputation system is absent from cloud industry, and introduce a *R3* reputation system tailored for cloud service delivery, by using reputation systems in e-Commerce for reference.

VI. CONCLUSIONS

Feedback rating-based reputation system is a promising way, to build trust between cloud users and cloud providers. However, nowadays, major cloud providers, e.g., *Amazon*, *Google* and *Microsoft* do not support such a reputation system, which hampers a cloud user from selecting a trusted cloud service before the service is executed. In view of this challenge, in this paper, we first study why reputation systems are absent from cloud providers. Afterwards, we put forward a novel reputation system CRS tailored to cloud service delivery. In the future, we will refine the proposed CRS reputation system by introducing more detailed and quantified reputation calculation formulas.

ACKNOWLEDGEMENTS

This paper is supported by the Open Project of State Key Lab. for Novel Software Technology (No. KFKT2012B31), Innovational Education Project for Postgraduate in Shandong Province (No. SDYY11138), Natural Science Foundation of Shandong Province of China (No.ZR2012FQ011, ZR2012FM023), Soft Science Research Project of Shandong Province (No. 2013RKB01040), DRF and UF (BSQD20110123, XJ201227) of QFNU.

REFERENCES

[1] M. Menzel and R. Ranjan, "CloudGenius: Decision Support for Web Server Cloud Migration", Proceedings of 21th International Conference on World Wide Web (WWW 12), ACM Press, Apr. 2012, pp. 979-988, doi 10.1145/2187836.2187967.

[2] V. Mareeswari and E. Sathiyamoorthy, "A Survey on Trust in Semantic Web Services", International Journal of Scientific & Engineering Research, vol. 3, Feb. 2012, pp. 1-5.

[3] N. Limam and R. Boutaba, "Assessing Software Service Quality and Trustworthiness at Selection Time", IEEE Transactions on Software Engineering, vol. 36, Jul. 2010, pp. 559-574, doi: 10.1109/TSE.2010.2.

[4] C. Rong, S. T. Nguyen, and M. G. Jaatun, "Beyond lightning: A survey on security challenges in cloud computing", Computers and Electrical Engineering, vol. 39, Jan. 2013, pp. 47–54, doi: 10.1016/j.compeleceng.2012.04.015.

[5] M. Armbrust, et al., "Above the clouds: A Berkeley view of cloud computing", Technical Report No. UCB/EECS-2009-28, University of California, 2009.

[6] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions", Journal of Systems and Software, vol. 86, Sep. 2013, pp. 2263– 2268, doi: 10.1016/j.jss.2012.12.025.

[7] J. Witkowski, "Incentive-Compatible Trust Mechanisms", Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 11), AAAI Press, Aug. 2011, pp. 1865-1866, doi: 10.1.1.222.1760.

[8] C. Fershtman and N. Gandal, "Migration to the Cloud Ecosystem: Ushering in a New Generation of Platform Competition", Communications & Strategies, vol. 85, Jan. 2012, pp. 109-124.

[9] EC2. http://aws.amazon.com/cn/ec2/ (accessed on 2013-9-9).

[10] CloudWatch. aws.amazon.com/cloudwatch/ (accessed on 2013-9-1).

[11] Rackspace review. http://www.rackspacecloudreview.com. (accessed on 2013-9-1).

[12] Amazon. http://www.amazon.com/(accessed on 2013-9-10).

[13] eBay. http://www.ebay.com/(accessed on 2013-9-10).

[14] R. Nadeau, E. Cloutier, and J.-H. Guay, "New Evidence About the Existence of a Bandwagon Effect in the Opinion Formation Process", International Political Science Review, vol. 14, Jun. 1993, pp. 203-213, doi: 10.1177/019251219301400204.

[15] S. M. Habib, S. Ries, and M. Mühlhäuser, "Towards a Trust Management System for Cloud Computing", Proceedings of 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 11), IEEE Press, Nov. 2011, pp. 933-939, doi: 10.1109/TrustCom.2011.129.

[16] M. K. Muchahari and S. K. Sinha, "A New Trust Management Architecture for Cloud Computing Environment" Proceedings of International Symposium on Cloud and Services Computing (ISCOS 12), Dec. 2012, pp. 136-140, doi: 10.1109/ISCOS.2012.30.

[17] S. Wang, Q. Sun, H. Zou, and F. Yang, "Reputation measure approach of web service for service selection", IET Software, vol. 5, Oct. 2011, pp. 466–473, doi: 10.1049/iet-sen.2010.0077.

[18] T. H. Noor, Q. Z. Sheng, S. Zeadally, and J. Yu, "Trust Management of Services in Cloud Environments: Obstacles and Solutions", ACM Computing Surveys, vol. 46, Oct. 2013, pp. 1-35, doi: 10.1145/2522968.2522980.

 ISBN: 978-1-61208-337-7

Web Services Framework for Wireless Sensor Networks

Mark Allen Gray and Philip Newsam Scherer
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County (UMBC)
mgray2@umbc.edu, pscher1@umbc.edu

***Abstract* — The recent proliferation of machine-to-machine service-oriented computing and the emergence of cloud computing platforms and services provides promising new capabilities for wireless sensor networks. A Wireless Sensor Network (WSN) by itself is heavily constrained to low-power usage resulting in low compute and storage capacity. Also, there are problems with aggregating sensor data from multiple WSN deployments for the purposes of creating and sharing sensor data in "big data" form and for sensor data fusion algorithm development. Researchers working on applications that require sensor data for modeling and prediction can simulate that data but testing their models against real-world sensor data and deploying their applications on real-time sensor data streams are repeating challenges. In this paper, we propose a web service framework that addresses and overcomes many of these common problems for users of WSNs. We describe the architecture of the framework and the REpresentational State Transfer (REST) Application Program Interface (API) for accessing framework resources. The results from our initial implementation demonstrated the framework operation over a continuous 175 hour data collection window and successfully presented statistics of processed streaming weather sensor data averaged over this entire data record.**

Keywords* — *Web Services; Service Oriented Architecture; SOA; Wireless Sensor Network; WSN; REST; Cloud Computing.

I. INTRODUCTION

The motivation for this research is the integration of wireless sensor networks with cloud services to operate on "big data' systems and provide access to computationally intensive compute resources. The fundamental requirements of the project were to create a web service that:

(1) Operates on big data,
(2) Provides a computationally intensive service,
(3) Hosts the data and compute resources in a cloud, and
(4) Implements a service oriented architecture.

Our approach was to meet these requirements by creating a web services framework for wireless sensor networks that addresses some of the challenges in that domain. A Wireless Sensor Network (WSN) by itself is heavily constrained to low-power usage resulting in low compute and storage capacity [1]. Also, there are problems with aggregating sensor data from multiple WSN deployments for the purposes of creating and sharing sensor data in "big data" form and for sensor data fusion algorithm development [2][3]. Researchers working on applications that require sensor data for modeling and prediction can simulate that data but testing their models against real-world sensor data and deploying their applications on real-time sensor data streams are repeating challenges [4][5]. Our web services framework (herein after referred to as the "framework") addresses these challenges.

In this paper, we first provide an overview of the framework in Section II and follow that with use case descriptions in Section III and related work in Section IV. We then describe the architecture of the framework and our initial implementation in Section V with a description of the results of our demonstration in Section VI. We conclude the paper with a description of future work in Section VII, a conclusion summary in Section VIII, acknowledgments in Section IX, and a list of references in Section X.

II. SERVICE DESCRIPTION

The web service that we provide is a framework for WSN data collection and processing in a cloud. The framework incorporates a service-oriented architecture (SOA) for distributed computing [6] and a REpresentational State Transfer (REST) [7] Application Program Interface (API) for machine-to-machine communication. To demonstrate the operation, a test case WSN is implemented and included as an example of using the framework. The primary components of the framework are:

(1) REST API
(2) REST Process Server
(3) Hyper Text Transfer Protocol (HTTP) Client Server
(4) Example Sensor Server
(5) Example Data Processing

The test case WSN used collects weather data from temperature, pressure, and humidity sensors. The sensor data is aggregated, time stamped, location stamped, and streamed into the framework where it is recorded to cloud storage resources and made available to users on-demand for inspection or for processing on cloud computing resources.

The entire system is illustrated in Figure 1. There are two basic types of users: data producers and data consumers. Data producers are users that deploy WSNs and add them to the system. When they add a WSN to the system they can choose to make the data recorded from their WSN private, shared in a group, or shared with the public. Data consumers are users that wish to consume data shared by the data producers. A data producer is, by default, a data consumer of their own WSN data and of any shared data from other data producers.

 ISBN: 978-1-61208-337-7

The REST process server forms the core of the system. It implements the API to all of the framework's web services, abstracting the services into a set of resources with operations on those resources and encapsulating all cloud resources comprising the framework. Access to the API requires an API key. An administrator that deploys and maintains a system that uses the framework will allocate an admin API key. Only users with the admin API key can add users to the system.

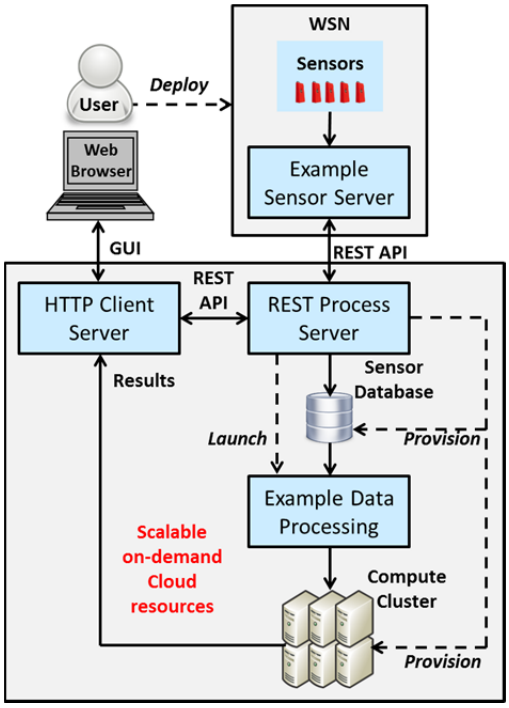


Figure 1. Framework System Components.

Users are added to the system through the HTTP client server. The HTTP client server implements a typical web portal Graphical User Interface (GUI) with user account signup and email verification which uses the admin API key to create the user. The HTTP client server is a user of the REST API. Once authenticated, users use their username and password to log in to their account. Each user has a user profile with an associated workspace and a dashboard for interfacing to the system.

Data producers will use their account to install their WSNs into the system. The account dashboard contains functions to add, modify, and remove a WSN. A WSN comprises a set of sensors and each sensor comprises one or more channels of data. For each WSN, sensor, and channel added to the system, the REST process server will allocate and return a Universally Unique IDentifier (UUID). The data producer will use their assigned API key and these UUIDs in their sensor server program for streaming their WSN sensor data into the system over the REST API to the sensor database. An example sensor server program written in Python [8] is included with the framework illustrating the use of the REST API for these purposes.

Data consumers will use their account to discover and use publically available WSN data or to subscribe to a group share. The account dashboard contains functions providing different views of WSN data including live sensor data being collected, recorded data in the sensor database, or the application of a data processing function to the data and a display of the results.

The system is currently designed with one built-in data processing function; an example data processing program is included. Future work will add the capability for data producers and data consumers to create a library of data processing functions and select the function to apply to a recorded dataset or live data. Additionally, the compute node type and number of nodes in the compute cluster running the data processing program will be user selectable.

All of the components in Figure 1 that are identified as "cloud resources" are deployed on a cloud platform. From the user's point of view, these resources are virtual and elastic. The elasticity of a cloud platform allows the system to scale up and scale down as demands require. For this project, these resources, due to schedule and budget constraints, were allocated on the UMBC BlueGrit computing system [9]. Future work will migrate the system to a commercial cloud platform for reliability and scalability testing purposes on a production cloud, for example Amazon Web Services (AWS) Elastic Compute Cloud (EC2).

III. USE CASES

The analysis, development, and deployment of wireless sensor network technologies are well-established in both academia and industry with applications in military, surveillance, environmental, industrial, transportation, healthcare, agricultural, home, and other many other use cases. Our framework extends these established use cases to address the following problems for hobbyists, researchers, and commercial enterprises:

(1) Aggregating sensor data from multiple WSN deployments,
(2) Creating and sharing sensor data in "big data" form,
(3) Providing a source of sensor data for sensor data fusion algorithm development,
(4) Replacing simulation data with real-world data in modeling and prediction algorithms, and
(5) Deploying algorithms against real-world real-time sensor streams in a cloud.

A. Hobbyists

WSN hobbyists could deploy the web services framework on a public cloud platform to manage the aggregation of their WSN generated data providing centralized access to their data from any Internet connected device. This would allow hobbyists to globally share their data with other hobbyists in a controlled system with authenticated users and managed access permissions. In addition to sharing data, hobbyists could share their sensor data processing functions and generally collaborate with each other on all aspects of their WSN interests. Public cloud platforms often offer free services for usage rates under

 ISBN: 978-1-61208-337-7

thresholds that would meet the requirements for the hobbyist use case.

B. Researchers

WSN academic, commercial, or military researchers creating intellectual property (IP) or other sensitive information could deploy the web services framework on a private (or community) cloud platform to manage the aggregation of their WSN generated data providing centralized access within their organization. This would allow the research team to collaborate amongst themselves or with other collaborative teams within their organization through a controlled system with authenticated users and managed access permissions. In addition to sharing data, researchers could share their sensor data processing functions and generally collaborate with each other on all aspects of their WSN research. As real-world data collects and builds in the sensor database, researchers across the organization could use the data in sensor data fusion and modeling and prediction algorithms.

C. Commercial

A production deployment of the web services framework on a commercial cloud platform could monetize the services and create value for the stakeholders. The service-oriented architecture is scalable over an elastic cloud infrastructure providing the service elasticity required for commercial service deployments. In this scenario, the cost to maintain the service scales up and down as the user demands scale up and down. Usage is on-demand with pay-as-you-go billing. Users on a commercial deployment could collaborate in the same way as described for hobbyists and researchers. The framework could be extended to support multiple cloud platforms with different price points that the user would choose or the user could provide the framework with the access credentials to cloud resources that they already have accounts with, in which case usage against those accounts would accrue against those accounts and a service fee would be added to monetize the transaction for the stakeholders.

IV. RELATED WORK

In this section, we look at current research and commercially deployed products that are related to web services for wireless sensor networks.

A. WSN Middleware

There is current academic research in the creation of WSN middleware primarily focused on the virtualization of WSN resources in a similar way that cloud computing offers virtualization of data and compute resources. One notable project is called “Serviceware” [10]. Serviceware is a service-oriented architecture of middleware that runs over the embedded WSN devices providing virtualization of the hardware in the form of services to multiple users concurrently. The motivation here is to drive down the cost of deploying, managing, and maintaining large-scale WSNs by maximizing the utility of the WSN resources to a broader user base and applications through infrastructure sharing. The authors note that maximizing WSN device utility also increases power consumption and further research is required to analyze the utility gains against the need to replace batteries more frequently.

B. SensorCloud

SensorCloud [11] is an existing commercially available proprietary product offering similar services as our web services framework for WSNs. Customers sign up for an account, choose a level of service with associated cost, receive an API key, and use the key to write code on their Internet connected sensor network devices that use their REST API. Like our REST API, users can get, add, update, and remove sensors and channels from their account and stream their sensor data to their account where it is stored in a database for query, retrieval, visualization, and analysis using data processing functions supplied by the user.

Unlike SensorCloud, our entire framework, including the front-end web portal and the back end REST server, will be open source and operate on top of open source web service software stacks. Additionally, our front-end web portal provides a user interface to get, add, update, and remove WSNs, sensors, and channels. For each resource added, a UUID is assigned and the user simply uses the UUID in their code. All of this can also be done through our REST API in the same way one would if using SensorCloud. Further, each WSN in our framework has Global Positioning System (GPS) location and altitude information and each sensor attached to a WSN has X,Y,Z grid coordinates relative to the GPS location and altitude. Streamed sensor samples include both time and location data supporting mobile wireless sensor networks. A feature that SensorCloud includes that we currently have not specified is the ability to define Short Message Service (SMS) and email alerts when certain user-defined conditions are detected.

C. Google’s Data Sensing Cloud

At the 2013 Google I/O Developer’s Conference in the San Francisco Moscone Center, Google implemented a version of the O'Reilly Data Sensing Lab, a collaborative project between O’Reilly Media and some of their partners. Google’s Data Sensing Lab deployed a 525 node, wireless sensor network at the conference feeding over 4000 continuous streams of sensor data into the Google Cloud Platform with Google Cloud Datastore for sensor data recording and Google Compute Engine for sensor data processing with results presented through a web application. Sensing consisted of temperature, humidity, noise, light, motion, and pressure to analyze the general atmosphere and traffic patterns of conference attendees throughout the conference’s changing of events and agenda. A Google representative at the conference stated “We think about data problems all the time and this looked like an interesting big data challenge that we could try to solve.” [12]

The fundamental architecture of Google’s project is very similar to our web services framework, although their focus was not in developing and demonstrating the required web services with an API, but on raising awareness and interest in hobbyists to build sensor nodes (the “lab” part of the project) and connecting to, and using, their cloud services.

 ISBN: 978-1-61208-337-7

D. Xively Cloud Services

Xively is building a business around services for the Internet of Things (IoT). User's develop and deploy their IoT products into the Xively "Connected Object Cloud" using Xively development tools, directory services, and data services through their API. The Xively API is a REST interface providing developers with web services to stream and record their sensor data to Xively servers and connect to other objects in the Xively cloud. Users, for a fee, can connect to those applications and embed the results in their websites or stream the data, for a fee, into their applications using the Xively API. The user relationships within this cloud ecosystem form a marketplace for real-time sensor fee-based data trading between connected devices and applications. "This sort of common platform is exactly what the Internet of Things really needs. Xively and similar platforms like Open.Sen.se will make it much easier and faster for unrelated devices to connect with each other and start delivering on the promise of smart homes, intelligent devices services and similar long-promised notions." [13]

An interesting capability here is the ability of sensor applications to use each other's data. The addition of REST services to our framework for the data processing function to use the data from other WSN sources would move our framework into the realm of this IoT paradigm. Whereas the Xively service is a closed proprietary deployment on Xively cloud resources, our framework, when deployed, will be open source for deployment on any cloud platform.

E. IBM InfoShpere Streams

Typical processing of big data resulting from sensor networks is performed on data that has been collected into a database where it is later queried, extracted, and analyzed. In the IBM InfoSphere Streams ("Streams") architecture, real-time sensor data streams are analyzed on a high performance computing platform before storing to a database. In this paradigm, data analysis is continuous, resulting in a continuous stream of low-latency real-time results for trend prediction, accelerating user responses to critical real-time events. A motivation for business applications is to address global economic competition; a motivation for government applications is to address global cybersecurity threats. Other example applications include telecommunications, financial services, healthcare, transportation, environmental, insurance, and utilities. Streams can consume data from satellites, sensors, cameras, news feeds, and a variety of other sources including traditional databases and Hadoop systems. In summary, Streams can process huge volumes and varieties of real-time data from diverse sources with very low latency, providing decision makers with the relevant and timely information they need [14].

An interesting capability here is the ability to process the sensor data in real-time before recording to a database. The addition of REST services to our framework for the data processing function to be a applied to the data either before or after recording to the database would extend the our framework to provide a similar capability. Whereas the Streams service is a closed proprietary deployment on IBM cloud resources, our framework, when deployed, will be open source for deployment on any cloud platform.

V. FRAMEWORK ARCHITECTURE

The top-level architectural components and interfaces comprising our web services framework for wireless sensor networks are illustrated in Figure 1 and listed here:

(1) REST API
(2) REST Process Server
(3) HTTP Client Server
(4) Example Sensor Server
(5) Example Data Processing

As described previously, there are two types of users: (1) data producers that deploy one or more WSNs and install them into the system for private, group, or public use and (2) data consumers that subscribe to and use WSN data shared by data producers.

The REST API forms the interface to the core services provided by the REST process server. The REST process server abstracts a set of resources that it manages and allows users to use those resources through REST request messages. All user API keys and cloud resources including the sensor database and compute cluster are allocated and managed by the REST process server.

The HTTP client server provides a web portal for users to create an account in the system and acquire an API key for using the REST API. The portal also implements a dashboard of functions for data producers to get, add, modify, and delete the resources representing their WSNs. Additionally, with their API key, users can perform these WSN administration functions directly from programs they may write.

The example sensor server runs on a WSN gateway node. It collects sensor samples from the attached sensor nodes and streams them to the REST process server where they are recorded to the sensor database. Users can access the live data or recorded data through the HTTP client server's web portal or from their programs. Live data and recorded datasets can be displayed. A user provided data processing function can be applied to datasets and the results displayed.

The test case WSN used in the project collects weather data from temperature, pressure, and humidity sensors. The sensor data is aggregated, time stamped, location stamped, and streamed into the framework where it is recorded to cloud storage and made available to users on-demand for inspection or for processing on a compute cluster.

A. REST API

The REST API forms the programming interface to the framework's set of web services. A RESTful [15] interface has client and server roles where clients initiate requests to servers and the servers process the requests and return responses. The requests and responses are formed into messages. The server manages resources that are addressable through the client requests. The representation of a resource and its state is captured in a document within the messages, typically in eXtensible Markup Language (XML) or

 ISBN: 978-1-61208-337-7

JavaScript Object Notation (JSON). Some REST interfaces support one or the other or both. The current specification of our framework uses the JSON format for the client request messages and the server response messages.

Requests and responses are typically processed using HTTP. Clients initiate requests using the HTTP request methods GET, POST, PUT, and DELETE. The client session transitions through its states based on the resource information returned from the server. This design pattern frees the server from the complexity of maintaining client and user interface state information, simplifying server logic and increasing server robustness, reliability, and scalability.

The framework's web services are abstracted into a set of resources where each resource is addressed by a unique Uniform Resource Locator (URL) and the operations on a resource are defined by the HTTP methods requested against the URLs. For GET methods, query parameters attached to the URL further define the data requested from the resource. For POST and PUT methods, the HTTP message body will contain the data to be transferred to the resource. The framework's root URL for accessing resources and services is: https://<FQDN>/api/<key>. FQDN indicates a Fully Qualified Domain Name, for example, www.example.com. Figure 2 illustrates the hierarchy of the resources comprising the framework and following is a description of each.

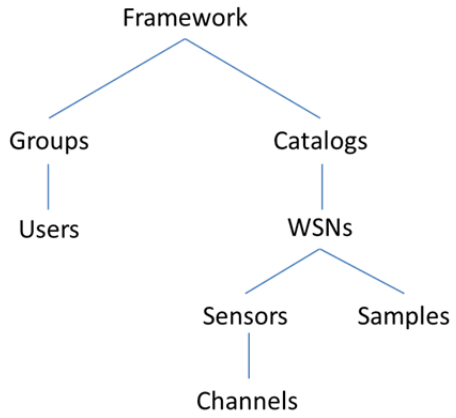


Figure 2. Framework Resource Hierarchy.

(1) Groups – A group resource is created by an admin user. A group is used to manage a group of users that share access to the same framework managed resources. Group services include listing all groups, creating and deleting a group, getting and updating a group's attributes, and adding and removing users to/from a group. The group resource was not implemented in the concept demonstration. The group resource URL relative to the root is: /groups/<groupid>.

(2) Users – Anyone with programmatic access to the REST API is a user. A program that sends a request message to the REST API must provide a user API key in the request message URL. API keys are created by an admin user that has an admin API key. The admin user makes a request on the REST API to create a user, and a unique user API key is returned. The HTTP client server web portal automates this process through the user account sign up process. User services include listing all users, creating and deleting a user, and getting and updating a user's attributes. The user resource URL relative to the root is:
/groups/<groupid>/users/<username>

(3) Catalogs – All resources offered to a user by the framework are organized into a hierarchy. At the top of this hierarchy is the catalog resource. The framework is currently architected with a single catalog, however, it can be extended to provide multiple catalogs for deployments that may wish to implement a "marketplace" of disparate catalogs distinguished, for example, by different legal agreements and terms and conditions of service. Catalog services include listing all catalogs, creating and deleting a catalog, getting and updating a catalog's attributes, and adding and deleting WSNs to/from a catalog. Catalog services were not implemented in the concept demonstration. The catalog resource URL relative to the root is: /catalogs/<catalogid>.

(4) WSNs – The top-level resource in a catalog is a wireless sensor network. A user that creates a WSN is considered a "data producer" and the "owner" of the WSN. The user can choose to make their WSN private, shared within a group of users, or shared with the public (all users). In the framework architecture the "sensor server" runs on a typical WSN gateway device where the WSN sensor devices stream their data to the WSN gateway for local storage or transmission over a network, in this case, transmission to the REST process server over the Internet. The WSN resource encapsulates information about the WSN gateway where the sensor server software will run. For example the initial GPS coordinates of the WSN gateway are specified when the WSN resource is created and they can be updated from time to time for a mobile WSN. WSN services include listing all WSNs available to the user, creating and deleting a WSN, getting and updating a WSN's attributes, adding and removing sensors to/from a WSN, recording samples to the WSN's sensor database, viewing live data from the WSN, viewing the WSN's recorded data, and applying a data processing function to a recorded WSN dataset. Applying data processing to the live stream was not implemented in the concept demonstration. The WSN resource URL relative to the root is:
/catalogs/<catalogid>/wsns/<wsnid>

(5) Sensors – The sensor resource encapsulates the identity, location, and sampling information about the sensor data channels that it physically contains and serves. Information about each sensor is captured when a sensor is created and it can be updated from time to time, for example the sampling frequency and the location for a sensor in a mobile WSN. Sensor services include listing all sensors owned by the user, creating and deleting a sensor, getting and updating a sensors's attributes, and adding and removing channels to/from a sensor. The sensor resource URL relative to the root is:
/catalogs/<catalogid>/wsns/<wsnid>/sensors/<sensorid>

(6) Channels – Sensor channels are the sources of sensor data in the framework. The channel resource encapsulates the data type and data unit for a sensor channel. For example a data type could be "Temperature" and the data unit could be "Celsius". Channel services include listing all channels owned by the user, creating and deleting a channel, and

 ISBN: 978-1-61208-337-7

getting and updating a channels's attributes. The channel resource URL relative to the root is: /catalogs/<catalogid>/wsns/<wsnid>/sensors/<sensorid>/channels/<channeleid>.

(7) Samples – A sample resource is the only resource that is not created by the framework (unless a simulation capability were added). A sample is the set of sensor channel values captured at any point in time, and location, by the sensor server according to the sample set configuration and the sample frequency. The sample set can be all of the sensors connected to the sensor server or a subset. The sensor server collects the sample set, adds a time stamp, adds a location stamp (GPS location and altitude and local grid location and altitude relative to GPS), and posts the data to a WSN.

B. REST Process Server

The REST process server component of the framework identified previously in Figure 1 is implemented in our concept demonstration with the software stack illustrated in Figure 3 below. At the top of the stack there are the framework's web services that we developed and integrated with the lower layers. The web services layer uses the services of the lower layers to implement all resources and services exposed by the REST API. Internally, it manages the sensor database and compute cluster for each WSN and schedules the data processing functions and returns results as directed by REST requests. Views of live sensor data, recorded sensor data, and processed sensor data are composed by the REST process server and returned in REST responses in JSON format.

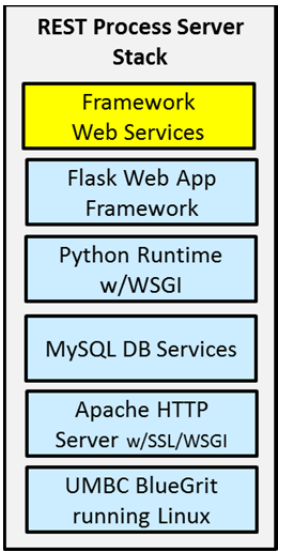


Figure 3. REST Process Server Stack.

For the concept demonstration, we utilized the resources of the BlueGrit computing platform at UMBC. The REST process server stack is built on Linux using open-source software. At the foundation is Apache HTTP Server. Secure Sockets Layer (SSL) encryption was enabled and utilized to secure all messages passing over the REST API. For database services, MySQL [16] was used. One database serves the framework and one database serves each WSN added to the system. Python was chosen for development of the framework's web services and Flask [17] was chosen to provide the required web application framework for deploying our REST web services. Flask is open source and implements the Web Server Gateway Interface (WSGI) 1.0 specification.

C. HTTP Client Server

The HTTP client server component of the framework identified previously in Figure 1 is implemented in our concept demonstration with the software stack illustrated in Figure 4 below. At the top of the stack is the framework's web portal that we developed and integrated with the lower layers. The web portal layer uses the services of the lower layers to implement the graphical user interface where users sign up for accounts, login into their account, and use a dashboard to manage their WSN deployments and to access and create views of WSN data and to process data and view the results. Internally, the web portal makes REST API requests to the REST process server on behalf of the user. The code that implements this interface is encapsulated in a PHP module that we installed into Drupal [18]. Drupal is a modular open-source Content Management System (CMS) framework written in PHP Hypertext Preprocessor (PHP).

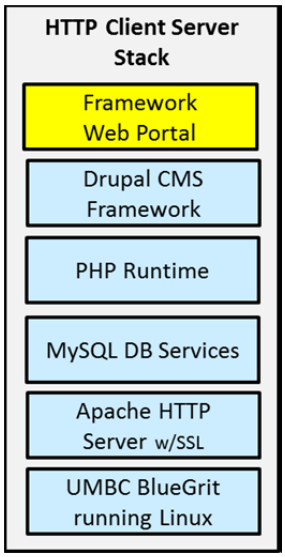


Figure 4. HTTP Client Server Stack.

For the concept demonstration, we again utilized the resources of the BlueGrit computing platform at UMBC for the HTTP client server, although there is no requirement that this server and the REST process server be on the same platform as long as they are both connected to the Internet. The HTTP client server stack is also built on Linux using open-source software. Also, at the foundation, is Apache HTTP Server. SSL encryption was enabled and utilized to secure all information passing between the user's web browser and the web portal. For database services, MySQL was used. A single Drupal database holds all the web portal

 ISBN: 978-1-61208-337-7

content and the module that we added to Drupal adds a table to that database.

The use of Drupal for the web portal immediately solves the problem of putting a "web face" on the service without reinventing all of the wheels that comprise a professional user-friendly dynamic website, which is a fundamental requirement that we have established for our framework. And because Drupal is modular, you install and use what you need. Only the core set of five Drupal modules (System, User, Node, Block, and Filter) and two contributed modules for enabling SSL, are required for the framework's web portal. Figure 5 is a screenshot of the web portal home page.

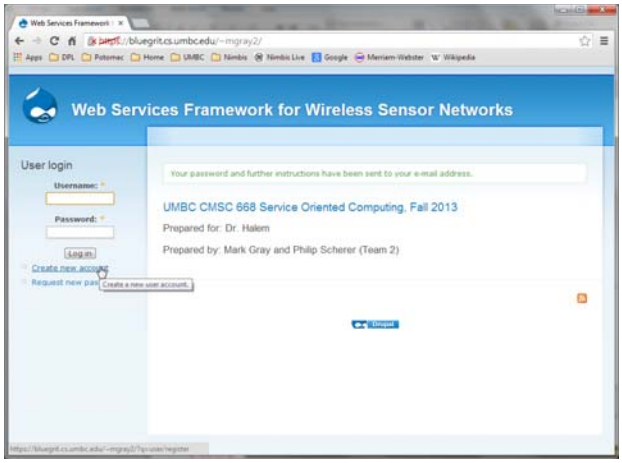


Figure 5. Web Portal Home Page.

The second and more important problem that Drupal immediately solves is the user signup and account management problem. User signup, authentication, login, and password management are entirely implemented with the core User module. When user authentication is complete and a user is created in the web portal, a REST request with the admin key is sent to the REST process server and an API key is allocated, returned, and made available to the user through their account on the web portal.

In addition to meeting these two fundamental requirements (a professional web face and user account sign up), a deployment of the framework's web portal could leverage the work from thousands of contributed Drupal modules, depending on the specific needs of the use case. For the hobbyist and researcher use cases identified previously, a deployment for these users could add profiles, forums, blogs, and other social networking tools for user interest discovery and collaboration. For the commercial use case, the open-source Ubercart [19] suite of Drupal modules could be added which comprise a complete end-to-end ecommerce workflow that integrates with several payment processing service providers. The commercial developers can create a catalog, add products, add terms and conditions, and build a shopping cart that buyers can take to checkout where their services are deployed. As of November 11, 2013 the Drupal developer community reached 30,000 with over 24,000 contributed modules [20].

D. Example Sensor Server

The example sensor server component of the framework identified previously in Figure 1 is implemented in our concept demonstration with the software stack illustrated in Figure 6 below. In this example test case, the sensor server is a "weather sensor server". This example is intended to be the "hello, world" for an initial test of a sensor server in a framework deployment. As such, it does not rely on actual physical sensors and the associated problems of procuring, installing, and getting the sensors to work just to test the framework. Instead, we rely on sensors that are already deployed with their data available to us on a REST API which we will inject into our system as if it were data collected on a deployed WSN. We used the Weather Underground (Wunderground) REST API [21] on a test account we setup that utilized their free level of service which was sufficient for both integration testing and the demonstration without exceeding the free usage levels.

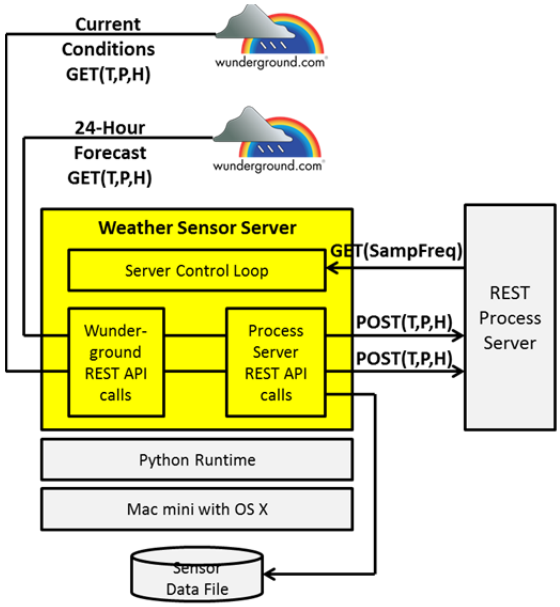


Figure 6. Example Sensor Server Stack.

Our weather sensor server was written in Python and executed on a Mac mini with OS X. A main control loop first updates its sampling frequency by a query to the REST process server where all WSN configuration parameters are maintained. It then sleeps for a time equal to the sample period. When it awakens it makes two calls to the Wunderground API for the current GPS location (in our demonstration the location is static): (1) the current weather conditions for the current location and (2) the 24-hour forecast for the current location. From the JSON responses, the temperature (T), pressure (P), and humidity (H) are extracted for both cases. The current(T, P, H) represent sensor 1 with three channels of data and the forecast(T, P, H) represent sensor 2 with three channels of data. Using the test user's API key and the resource IDs assigned by the web portal, two REST requests are made to the REST process server: (1) a POST of the current(T, P, H) with a timestamp

 ISBN: 978-1-61208-337-7

equal to the current time and (2) a post of the forecast(T, P, H) with a timestamp equal to the current time plus 24 hours. In our demonstration, we set the sample frequency to 12 times per hour, or once every five minutes and collected data for 151 hours. In addition to pushing the data out on the REST API, we also logged it to a text file in Comma Separated Value (CSV) format for testing and integration.

E. Example Data Processing

The example data processing component of the framework identified previously in Figure 1 is implemented in our concept demonstration with the program illustrated in Figure 7 below. In this example test case, the data processing program is a "weather data processing" program. Like the example sensor server, this example is intended to be the "hello, world" for an initial test of a data processing program in a framework deployment.

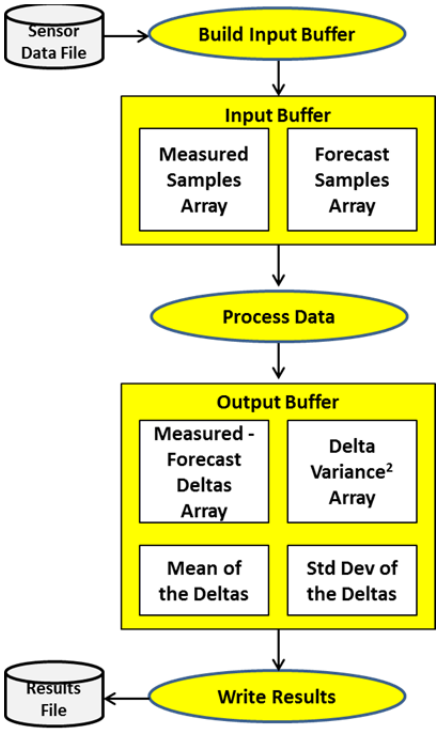


Figure 7. Example Data Processing Program.

The input to the data processing program is a sensor data file containing a dataset extracted from the sensor database. In a completely integrated system, the sensor data file would be pulled from the sensor database by the REST process server after a request to apply the data processing program to the data. The REST process server would place the file into a shared file system available to a compute cluster and launch the data processing program on that cluster. In our concept demonstration, we took the sensor data file that was recorded on the sensor server and uploaded it to a BlueGrit compute blade and executed the data processing program on the data.

The data processing pipeline shown in Figure 7 illustrates a general input -> process -> output dataflow. For this test case, the sensor data file contains 175 hours of data collected on two sensors with 12 samples collected per hour per sensor. Each sample contains a timestamp and the current temperature, pressure, and humidity for that sensor. Each sensor 1 sample contains the actual temperature, pressure, and humidity at that time. Each sensor 2 sample contains the 24-hour forecasted temperature, pressure, and humidity for that time. The "build input buffer" function averages the 12 samples for each one hour time slot and creates two arrays indexed by hour; the "measured samples array" from the sensor 1 samples and the "forecast samples array" from the sensor 2 samples. The data in the input buffer is then processed.

The "process data" function consists of four computations with the results of each computation saved to the output buffer. They are:

(1) Compute the deltas between measured and forecasted
(2) Compute the arithmetic mean over the deltas array
(3) Compute the delta variance^2 array
(4) Compute the standard deviation

The "write results" function summarizes and formats the contents of the output buffer and writes it to a text file where the REST process server picks it up.

Note that the first 24 hours (hour 0 through hour 23) of the 175 hours of data collected have no 24-hour forecast values for comparison and the last 24 hours (hour 151 through hour 174) have no measured values, therefore, the actual computable dataset is 127 hours of data from hour 24 to hour 150.

VI. PROJECT RESULTS

We successfully completed the initial design and implementation of each framework system component and demonstrated the functionality of each component separately, with partial integration of the HTTP client server with the REST process server. The following demonstrations of the REST API specification were presented:

(1) User account creation
(2) User dashboard walkthrough
(3) Sensor server demonstration (of live data)
(4) Sensor data processing (of recorded data)

Implementation consisted of installing and configuring the open-source components of the server stacks and software development of key components. The web services in the REST process server stack consisted of a Python application that we developed (about 1000 lines) and installed on Flask. The web portal in the HTTP client server stack consisted of a PHP module (about 1100 lines) that we developed and installed in Drupal. The example sensor server consisted of a Python application that we developed (about 130 lines) and installed on an Internet connected Mac mini. The example data processing program consisted of a C program that we developed (about 900 lines) and executed on a BlueGrit compute blade. The computation results of the data processing program are presented in Figure 8. A performance comparison between a Windows laptop and a BlueGrit blade is presented in Figure 9.

 ISBN: 978-1-61208-337-7

```
Data window begin time       = Tue Nov 26 12:06:49 2013
Data window end time         = Tue Dec 03 18:40:59 2013
Data window hours total      = 175
Forecast range (hours)       = 24
Hours of data processed      = 127
Number of samples processed = 9144
Mean of Temperature Deltas   = -0.731508 Fahrenheit
Mean of Pressure Deltas      = -0.013176 inHg mslp
Mean of Humidity Deltas      = 3.023204 %rH
StDv of Temperature Deltas   = 2.135467 Fahrenheit
StDv of Pressure Deltas      = 0.046519 inHg mslp
StDv of Humidity Deltas      = 8.797242 %rH
```

Figure 8. Data Processing Computation Results.

Windows 7 Laptop, Intel Core i7 @ 2.7Ghz, 8GB RAM	Performance on UMBC BlueGrit Intel01 Blade
= 0.026 seconds	***= 0.015 seconds***
= 2.84 usecs/sample	***= 1.64 usecs/sample***

Figure 9. Data Processing Performance Results.

VII. FUTURE WORK

Looking at this project as a set of sequential phases, this initial phase represents a three month concept study culminating in a concept demonstration which we have documented in this paper. Future work would address incomplete areas in the concept study and include research on new capabilities.

A. *Incomplete areas to address*

- Complete the integration of the concept study components
- Demonstrate parallel computation on the sensor data
- Implement the group resource and services
- Demonstrate on a commercial public cloud (Amazon)

B. *New capabilities to research*

- Cloud scalability, elasticity, load balancing
- Mobile WSN demonstration
- Data processing library creation and sharing
- Virtualization middleware on the sensor server and sensors
- Data processing on multiple input sources and types
- Data processing on live sensor streams
- Compute instance type and cluster size selection
- Network Protocol Time (NTP) on the sensor server

VIII. CONCLUSION

In conclusion, we successfully demonstrated an architecture and initial implementation of a web services framework for wireless sensor networks. A test case WSN was simulated on a Mac mini pulling actual real-time weather data from the Wunderground REST API and feeding it into the framework. Each framework component was individually constructed, tested, and demonstrated. The cloud storage and compute resources were provisioned from the UMBC BlueGrit computing platform. Future work includes end-to-end integration and testing of all components, the demonstration of parallel computation, the implementation of user groups, and a demonstration on a commercial public cloud. Other future work includes several areas of research, notably cloud scalability, mobile WSN, data processing libraries and sharing, and virtualization middleware extending the concept of cloud computing into the WSN domain.

IX. ACKNOWLEDGMENT

The authors thank Dr. Milton Halem for his direction and encouragement over the course of this project and his teaching assistant Lawrence Sebald for his support in the installation and configuration of various software services on the UMBC BlueGrit computing platform.

X. REFERENCES

[1] W. Dargie and C. Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice, Wiley, 2010.

[2] A. Cuzzocrea and G. Fortino, “Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-Of-The-Art and Future Research Directions”, 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, pp. 583-588.

[3] R. Govindan, J. M. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, “The Sensor Network as a Database”, University of Southern California, 2002, pp. 1-8.

[4] D. Tracey and C. Sreenan, “A Holistic Architecture for the Internet of Things, Sensing Services and Big Data”, 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, pp. 546-553

[5] S.K. Dash, S. Mohapatra, and P.K. Pattnaik “A Survey on Applications of Wireless Sensor Network Using Cloud Computing”, International Journal of Computer Science & Emerging Technologies, Volume 1, Issue 4, December 2010, pp. 50-55.

[6] M. P. Singh and M. N. Huhns, Service Oriented Computing: Semantics, Processes, and Agents, Wiley, 2005.

[7] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Dissertation, University of California, Irvine, 2000.

[8] Python. [Online]. Available: www.python.org. Retrieved Dec 2013.

[9] BlueGrit. [Online]. Available: http://bluegrit.cs.umbc.edu/userdocs.php . Retrieved December 2013.

[10] S. Rea, M. S. Aslam, and D. Pesch, “Serviceware - A Service Based Management Approach for WSN Cloud Infrastructures”, 10th IEEE International Workshop on Managing Ubiquitous Communications and Services 2013, San Diego, March 2013, pp. 133-138.

[11] SensorCloud. [Online]. Available: http://www.sensorcloud.com/system-overview. Retrieved Dec 2013.

[12] K. Fogarty, “Google's Wireless Sensors: Big Data or Big Brother?”, www.networkcomputing.com, May 22, 2013. Retrieved Dec 2013.

[13] B. Proffitt, “Xively Actually Connects Things to the Internet of Things”, www.readwrite.com, May 14, 2013. Retrieved Dec 2013.

[14] R. Rea, “IBM InfoSphere Streams, Redefining real-time analytics processing”, IBM Software, Thought Leadership White Paper, May 2013, pp. 1-8.

[15] L. Richardson and S. Ruby, RESTful Web Services, O'Reilly Media, 2007.

[16] MySQL. [Online]. Available: www.mysql.com. Retrieved Dec 2013.

[17] Flask. [Online]. Available: http://flask.pocoo.org. Retrieved Dec 2013.

[18] Drupal. [Online]. Available: https://drupal.org. Retrieved Dec 2013.

[19] Ubercart. [Online]. Available: http://www.ubercart.org. Retrieved Dec 2013.

[20] S. Choudhury, “30,000 Developers in Drupal.org and growing…”, [Online]. Available: https://drupal.org/node/2133153. Retrieved Dec 2013.

[21] Weather Underground (Wunderground) API. [Online]. Available: http://www.wunderground.com/weather/api/. Retrieved Dec 2013.

 ISBN: 978-1-61208-337-7

An Ontology for User Profile Modelling in the Field of Ambient Assisted Living

Carina Fredrich, Hendrik Kuijs, Christoph Reich

Faculty of Computer Science
Furtwangen University of Applied Science
Furtwangen, Germany
Email: {Carina.Fredrich, Hendrik.Kuijs, Christoph.Reich}@hs-furtwangen.de

***Abstract*—The lack of social integration of elderly people, especially with impairments like restricted mobility, is a huge problem. Often, these people become isolated and social contacts become impoverished. Ambient Assisted Living (AAL) IT systems should support elderly people to stay in contact with their social environment and should be adaptable exactly to their personal needs. In this paper, we present a platform offering assistance in communication, information acquisition and learning for elderly people to allow them to stay longer at their own familiar homes. The services of this platform are context aware and personalizable. Many AAL systems are context aware, but often focus on the environmental context and not on the users themselves and their personal characteristics, like health condition, interests, needs, etc. In this work, the context is modelled as an ontology, where the user is the central concept of the platform, in order to realize personalization of services and a better assistance by the system. The ontology developed by the project Person Centered Environment for Information, Communication and Learning (PCEICL) offers a historical view of the user's changing characteristics and environment, is simply expandable and is used within the platform by software agents to communicate between single services to adapt to the users needs.**

Keywords–Ontology, Context, User Centric Ontology, AAL, PaaS, OSGi, JADE, software agents

I. INTRODUCTION

Many solutions in the field of Ambient Assisted Living (AAL) solve problems of home automation, freedom of barriers and emergency diagnosis, as stated in [1], [2] or [3], to allow elderly people to stay longer at home. But when they need, for example, every day assistance, when they are suddenly mobility-disabled, there is the problem of social contact depletion. Especially in rural regions, it is not that simple for older people to keep up their social contacts if they are physically limited. Ordinary things like meetings with friends, family members, or club members or going shopping aren't possible any more. This leads to loneliness, isolation and often mental health problems. To counteract this, personalizable systems, which help them stay informed or assist them in their communication, are needed, especially if they are adaptable to the needs of the elderly people, as discussed by S. Lauriks et. al. [4].

For the personalization of a system and its services, it is necessary to integrate context awareness. The system must have knowledge about the user's interests, preferences, impairments, capabilities, etc. Most of the actual AAL systems consider only the user's environment (e.g., temperature, location, smoke, etc.), but do not see the user as a central element. A step towards this, the person's needs are taken into account and bear in mind the environmental information of the user to get a personalizable AAL platform.

The research project ***P**erson **C**entered **E**nvironment for **I**nformation, **C**ommunication and **L**earning (PCEICL)* targets the development of an age-appropriate platform, which assists the users in their daily tasks of information acquisition, communication and learning in order to live at home as long as possible and at the same time stay socially integrated. Because of the impairments emerging in advanced age, it is absolutely necessary to adapt the user interface but also the service functionality to the needs of each single user. The personalization is important because of the different combinations of impairments and capabilities. The PCEICL platform offers personalized services, which use context information about the user and the environment to adapt according to the needs of the user.

In this paper, context is used as defined in A. Dey and G. Abowd in [5]:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

There are other definitions only considering the environment of the user, e.g., the location, other people and resources nearby like Schilit et al. in [6]. But for reaching personalized assistance, it is essential to centralize the user's needs and include the actual user's environment. Because A. Dey and G. Abowd determine context as both, i.e., the user and his environment, the definition above is chosen as a basis for this paper.

Section II presents related work showing the different notions of context and context awareness. The use cases of the AAL platform described in Section IV, are shown in Section III. The platform for personalized services uses an ontology for context modelling (see Section V), which is demonstrated in action in Section VI. In Section VIII, conclusions are drawn and future work is shown.

II. RELATED WORK

Due to the relevance of the AAL topic, there are many projects, which develop assistance systems for elderly people helping them to stay longer at home. Most of them use

 ISBN: 978-1-61208-337-7

some kind of context information to make their systems more intelligent and adaptable. Often in these projects, context is used to define the environment of the user and not the user himself. This is because the majority of the projects develop home automation systems or emergency detection systems. But only a few aim to offer an assistance in staying socially integrated.

The *SOPRANO (**S**ervice **O**riented **PR**ogrammable sm**A**rt enviro**N**ments for **O**lder Europeans)* project [1], for example, developed an open middleware for AAL solutions. The *SOPRANO Ambient Middleware (SAM)* receives user commands or sensor data, enriches them semantically and determines an adequate system response, which is then performed by the connected actors installed in the living environment. If, for example, SAM receives the information that a window is open, it analyses the remaining context information and can inform the user about the open window, before he is leaving the house. The components communicate over semantic contracts and are based on a common domain ontology. This ontology is designed state-driven, that means that every concept (device, person, location, etc.) of the ontology is represented by its actual state. The PCEICL platform, on the other hand, focuses on the user. The most important is to describe the user, since for information retrieval the user's condition is essential. So it is not useful to apply the SOPRANO ontology, which focuses on the sensors and actors, i.e., the environment of the user.

Another example considering the environment of the user is PersonisAD presented by M. Assad et al. in [7]. Some information about the user like his preferences are also part of the consideration of the PersonisAD framework, but aren't detailed enough to reach a good personalization for older people, like the PCEICL ontology.

The ***UNIVER**sal open platform and reference **S**pecification for **A**mbient **A**ssisted **L**iving) (universAAL)* project [8] aims to join different approaches from lots of projects to a unique AAL solution. One of this included projects is SOPRANO [1]. The goal of universAAL is a platform, that makes it viable to develop AAL services. For this, there will be developer tools, a store for distributing AAL services and a runtime environment. So all of the stakeholders will be supported. The universAAL platform is based on OSGi [9] and ontologies are used as a common language for the components, too. Because of its goal to create a standardized AAL solution, it is possible that the universAAL platform and its ontologies will be building a solid base for future work in the PCEICL project. But, at the moment, the universAAL project is still in progress.

MobileSage aims to develop a smart phone based help-on-demand service [10][11]. It means that the smart phone offers context aware, personalized and location aware services supporting the independence of elderly people. Such services could support the navigation, the handling of devices like ticket vending machines or household appliances or other daily tasks. The personalization and context awareness is realized by an ontology, which considers not only the environment of the user but also the user and his characteristics. It is one of the few ontologies in the field of AAL, which models a user profile and the environment of the user. The central concept of the ontology is the user, who is described by his profile. The user profile therefore is divided in subprofiles like a preference profile, a health profile or an interest profile. But for the help-on-demand services the focus is still on the environment of the user to offer, for example, services depending on the location of the user. The PCEICL project places greater emphasis on the user, who has to stay at home and isn't mobile any more. For this, the user must be described more in detail in his health condition to provide him optimal assistance with the daily tasks. Overall, both ontologies describe a user profile and so there are many similar concepts but PCEICL concentrates on information retrieval, communication and learning.

Another ontology that models a user profile for ambient assisted living services is *AALUMO*, which is presented by P.A. Moreno, M.E. Hernando and E.J. Gómez in [12]. *AALUMO* extends the *General User Model Ontology (GUMO)*, which is an ontology for general use in many domains and scopes. So it is not adapted to the special characteristics of elderly users. GUMO is shown by D. Heckmann et al. in [13] and describes the user in detail from the heart beat or the emotional state to the interests or the personal information. AALUMO added concepts like chronic diseases, which is a composition of GUMO concepts (disease, physical and psychological limitations, medication, etc.) for better adaptation to the properties of the elderly users. In the PCEICL ontology, only the information that is really necessary to customize the services in an optimal way will be saved. Additional information about the user, which is not yet covered by the ontology but will be required in the future, could be added easily.

There is also a user profile ontology presented by M. Sutterer in [14], but it is not adapted to the special needs of elderly people. The personalization of service is situation-dependent and therefore the user needs to indicate special preferences for each situation, which are considered when the situation occurs. It is assumed that this is hard to be realized by elderly people and is therefore not the aim of the PCEICL project.

To summarize, the PCEICL ontology is based on concepts of the ontology of *MobileSage* and there will be future developments based on the results of *universAAL*.

III. USE CASE

In this section, the use cases are described to show the usage of the context aware PCEICL platform and show the usefulness of the PCEICL ontology in the evaluation section (see Section VI). First, a brief description of the user is given showing his current situation.

User description: *Mr. F. is a 73 years old widower, has two children and lives alone on a big farm outside a small rural town. One of his children lives abroad, the other in a city far away, so they rarely can visit their father. Mr. F. is a member of a model railway club in the city about 30 km away. Once a week, he used to attend the club meetings. Recently, he broke his leg after he slipped on an icy surface and therefore has mobility limitations and cannot leave the house very often. His children are worried, because his friends and also his club mates from the model railway club don't have much time to visit Mr. F. personally.*

To support Mr. F. in his daily tasks, the PCEICL platform assists him in communication, information acquisition and learning. With this platform, he is able to stay socially integrated. Especially his children can feel more comfortable, by

 ISBN: 978-1-61208-337-7

contacting him more easily during this convalescence period. Next, the use cases for each of the three main support areas are described.

A. Use Case (Communication Assistance)

Each contact in Mr. F.'s address list has specified several communication channels, like telephone, email, video chat, SMS, etc. Additionally, each contact has defined information about the availability of the different communication channels based on the contact person's daily habits or appointments. It is assumed that there is always at least one communication channel, where the contact can be reached in an urgent situation. Suppose Mr. F. wants to communicate with one of his children. All he has to do, is to select the name of his children from his address list. After this, the PCEICL platform automatically selects the communication channel based on the aforementioned schedule and preferred communication channel (e.g., SMS because his son is busy).

B. Use Case (Information Acquisition Assistance)

Another application on the platform helps Mr. F. find assistance from other people in the countryside. Although he stopped active farming, there are many things to be done on his farm. He has to feed his hare, do lawn mowing, do shopping, repair things once in a while, clean the house, etc. All this is very difficult or impossible for a mobile restricted person. If he needs help, he should be supported by a PCEICL service. Because the platform knows the user's health condition, it offers an "search-and-offer" service automatically. This could be used to search assistance in the aforementioned daily or weekly tasks, but also for special occasions. For instance, if there is a social meeting with the model railway club, he would like to participate in, the system will automatically help him to get a lift or special transportation. If his health condition gets worse and he cannot attend the meeting, the lift will be canceled by the system. Because the system also knows about the environment, e.g., the weather condition, it can automatically organize help, for example, to clear the snow.

C. Use Case (Learning Assistance)

Every day Mr. F. uses the PCEICL platform's fitness service. The fitness service guides Mr. F. through his everyday exercises like arm circles, arm curls or leg straightening. Because of the new change in health condition the service automatically skips exercises which are not suitable for a broken leg and adds some arm movement exercises to reach the same fitness level. As he recovers slowly from his leg fracture, the system can include specific exercises for his legs to restore his mobility.

The use cases described above are basis for the evaluation section (see Section VI) to show the benefits of the newly developed PCEICL platform and ontology.

IV. The PCEICL Platform

Figure 1 shows the PCEICL platform realized as an OSGi Platform as a Service (OSGi-PaaS) inside a cloud infrastructure to benefit of the PaaS scalability and the simple extensibility through the OSGi bundle mechanism. The scalability

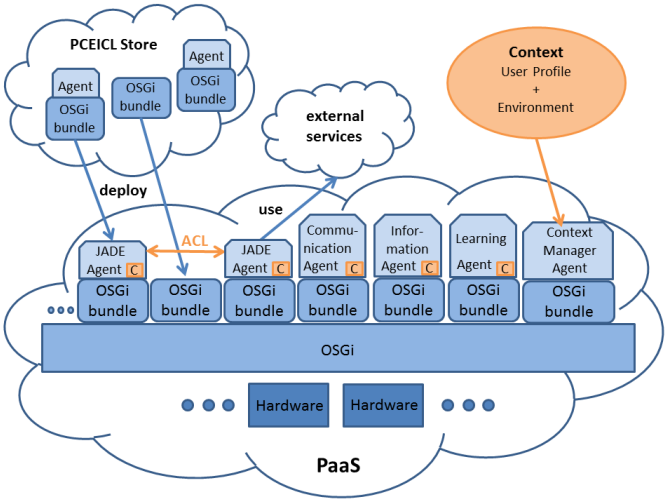


Figure 1: The PCEICL platform

is needed for CPU intensive functionalities, like image and speech recognition. More and more OSGi bundles in the field of home automation and health care are appearing and can be easily integrated [15]. A specific OSGi bundle of the PCEICL platform is a software agent OSGi bundle, which facilitates the development of intelligent behavior and realizes standardized communication between agents by using Agent Communication Language (ACL) [16]. ACL defines the use of ontologies, which we realized with the PCEICL ontology (as shown in Figure 1). For the software agents, the Java Agent DEvelopment Framework (JADE) [17] has been chosen. JADE is widely used and already provided as an OSGi bundle, which makes an integration of the agents in the OSGi environment straightforward.

Agents are well known, when developing intelligent systems, for their support for standard communication between them, and for helping the integration of external services. External services could be existing services like nursing services, communication services to contact a doctor, weather services, etc. For example, the user personalized platform could use the user information to exchange data between the PCEICL platform and the nursing service's system to indicate the necessity of a visit to the elderly person.

The distribution, installation and deployment of the PCEICL services is realized by an OSGi bundle store (PCEICL Store, see Figure 1) equivalent to modern app stores for smart phones or operating systems. These services can be selected by the elderly users, by a personal assistant, by a relative or by the system itself, to automatically update services or suggest new services to be installed to satisfy the needs (interests, impairments, etc.) of the user.

The profile and environment of the user is modelled by the PCEICL ontology. This ontology includes all the information about the person and its personal relevant environment to adapt the PCEICL services and applications (e.g., impairments, interests, hobbies, etc.). The stored profile of the user is managed by the PCEICL platform in the form of a *Context Management Agent* (see Figure 1), deciding which service is getting what kind of information about the user. Because

 ISBN: 978-1-61208-337-7

of the privacy data minimization principle, not every service should get all information. For example, the communication service providing audio and video telephone calls should not get information about the health condition of the user. It gets access to information about the volume, the contacts and possible colours or font sizes for the user interface.

The needs, interests, health condition, etc. are gathered in several ways. On the one hand, there is a manual acquisition of user data during the initialization of the system by the user or a personal assistant and on the other hand there is a continuous analysis of context data through, e.g., sensors resulting in the adaption of user and environment data. The environment context is managed by the *Context Management Agent*.

In Figure 1, there are special agent OSGi bundles for the three support areas of the PCEICL platform: communication, information and learning. These agents realize an assistance of the user with the context information given by the context management agent.

V. The PCEICL Context Ontology

For an optimal personalization of the PCEICL system, the user context must be modelled, a semantically underpinned agent communication is needed, the information should be semantically connected to each other and there should be rules describing the usage of the single information. As a result, the context information and the ACL ontology part is modelled in the form of the PCEICL ontology, which enables the description of information relationships and the deduction of new data out of existing information. For example, if the system knows about the health condition of a user, capabilities and impairments could be deduced from it. This ontology is used in the Foundation for Intelligent Physical Agents-Agent Communication Languages (FIPA-ACL) [18] and is managed by the context manager agent of the PCEICL platform.

The procedure of the ontology design for the PCEICL platform is based on the approach of N. Noy und D. McGuinness [19]. The ontology is developed iteratively and, for this reason, there will be adjustments of the ontology in the whole lifecycle of the PCEICL project. Automatic updates of the concepts or just the individual elements of the ontology must be possible.

The PCEICL ontology is the base for saving and interpreting context information. It specifies the way of describing the user, his properties and his environment for all components of the PCEICL platform, particularly for the personalized services. The ontology describes primarily the user and his properties but additionally the user's environment like weather, time, date, devices/sensors, etc. Figure 2 shows an overview of the basic concepts of the ontology. A more detailed description can be found in [20].

In addition to the mandatory user data, like the personal information (name, address, date of birth, contact information, etc.), there are some more specific and more complex concepts, like the interests, preferences, capabilities, or the health condition. Many of the ontology classes are fixed defined classes, defining the properties an individual of this class must have. This leads to a better consistency of data.

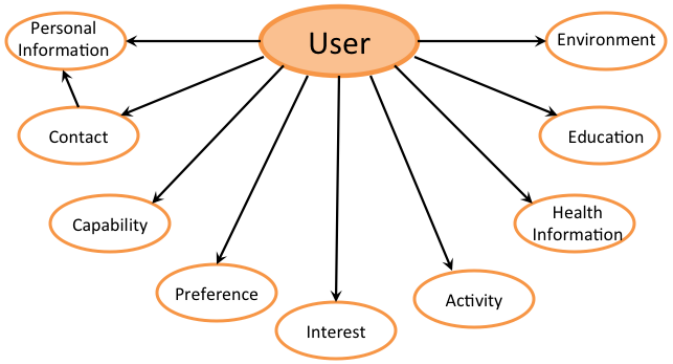


Figure 2: Overview of the basic concepts of the PCEICL ontology

The central role of the user is reflected in the class *User*, as shown in Figure 2. It is connected via properties to almost every other main class of the ontology. The *User* class is derived from the defined class *Person*, shown in Figure 3. For all instances of this class it is necessary to have exactly one id and exactly one personal information. So, every person is defined uniquely in the system.

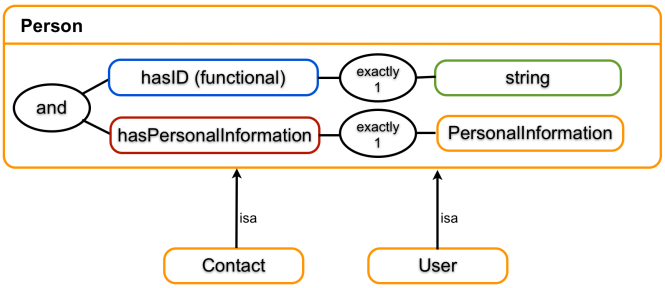


Figure 3: Class *Person*

For the PCEICL communication services of the platform, it is necessary to have a contact list. Every *Person* in this list is represented by an instance of the class *Contact*, which is also derived from class *Person*. Additionally to an ID and the personal information, there is a categorization of each contact to give the system knowledge about the relationship of the contact to the user. So, the system could distinguish between family members and, for example, doctors. For each contact, photos could also be saved. The information about the communication channels of the persons are saved in the concept *PersonalInformation*.

The class *Education* contains only information about the user's academic status, e.g., foreign languages the user speaks. This class can also be used to save data about learning progresses of the user. More information about the education of the user is at the moment not relevant for the platform and therefore, this information should not be saved. If it will be relevant in future to save more information, the ontology could easily be extended by other attributes.

The interests are represented by the class *Interest*. To differentiate between interests, like sports watching on TV and actively exercised interests, like actively playing a sport, the class *Activity* is also part of the ontology. Figure 4 shows

 ISBN: 978-1-61208-337-7

the concept of the class *Interest*, which is almost the same as for the class *Activity*. In addition to the name and the type of the interest or activity, it is possible to store more specific information, like pictures, websites, or descriptions (class *AdditionalInformation*). The class *Interest* should also have a property *hasLevel*, which describes how much the user is interested in something.

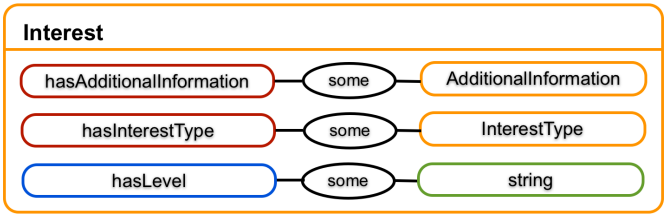


Figure 4: Class *Interest*

In the class *Preference*, distinction is made between *Personal Preference*, which is structured like the class *Interest* and *System Preference*. Personal preferences could be, for example, the favorite color or the taste of music. In the class *System Preference* settings of services or the platform itself could be saved. This settings could be derived from other information. For example, there could be a user who cannot differentiate between red and green. As a result, the system should not use red and green for the user interface. In such cases, the system should automatically derive the system preferences from the health condition or other information, which are given about the user. In the system preferences, account information is also stored, which can be username and password, a certificate, etc.

The class *Capability* is divided into *CognitiveCapability* and *PhysicalCapability* and is also structured like the class *Interest*. The capabilities and limitations stored in this class should also be derived from the health information of the user.

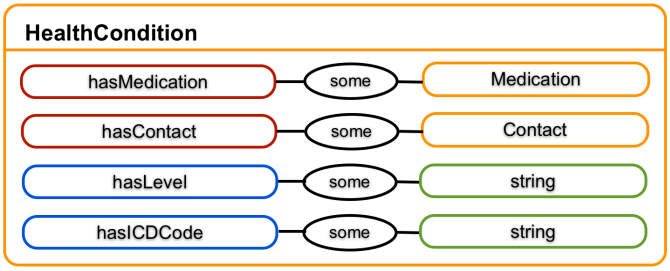


Figure 5: Class *HealthCondition*

The most important information is the health condition of the user. A lot of other information could be generated knowing about the impairments of the user. Therefore, a class *HealthCondition* exists, which is presented in Figure 5. Every single instance describes the impairments and diseases of the user. So, information can be saved about the medication of the disease or the physician treating the disease, who is saved as a contact in the list. Each impairment or disease can be rated by the property *hasLevel* by saving information such as the dioptre numbers or the status of disease. For the correct medical description of the disease, you can save an *ICD-Code* for each instance of the class *HealthCondition*. ICD means *International Statistical Classification of Diseases and Related Health Problems Code* [21] and is a worldwide used coding of diagnoses. So, a uniform and correct description of the health condition of the assisted person can be reached.

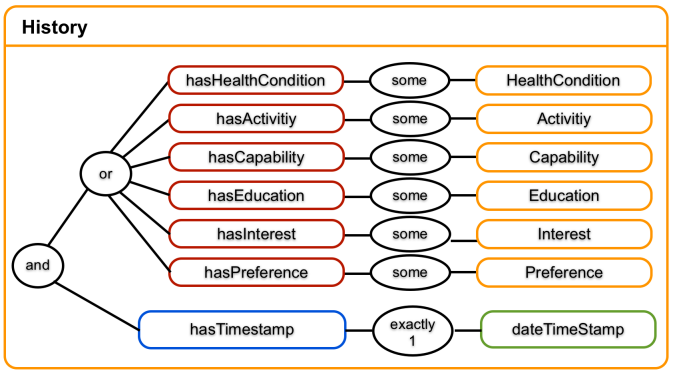


Figure 6: Class *History*

The PCEICL ontology also offers a concept to consider the context historically. So, you can observe the changes of the learning behaviour or of the capabilities and impairments, for example. Therefore, a class *History* is designed taking an expired instance of one of the classes shown in Figure 6 and combining it with an actual timestamp. For example, if the user has a cold, this information will be saved as *HealthCondition* instance. If the cold is overcome, the instance will be combined with a timestamp and will be saved as a history object.

A. PCEICL Ontology Example

Figure 7 shows an example of the user profile information covered by the concepts of the ontology. It represents the user profile of Mr. F., who has been introduced in the use cases of Section III.

VI. PCEICL Ontology in Use

In this section, the introduced use cases of Section III are called into play to show the platform services using the concepts of the ontology. The development of some parts of the PCEICL platform is still in progress. When the prototyping phase is completed, there will be a socio-scientific accompanying evaluation of the services by a group of persons aged 60 and older.

A. Use Case (Communication Assistance):

The communication app itself is just a simple client service getting only the information needed to interact with the user. Therefore, the main context information is about the contacts saved as individuals of the class *Contact*. For the communication app, information about the system preferences of the user are used to adapt the GUI. Most of the settings can automatically be derived from the health condition of the user (ontology class: *HealthCondition*). For example, if the user has a red-green colour blindness, these colours should be excluded from possible system settings or if the user has a hearing deficiency, the volume should not be set under a minimum level. These pieces of information could be saved

 ISBN: 978-1-61208-337-7

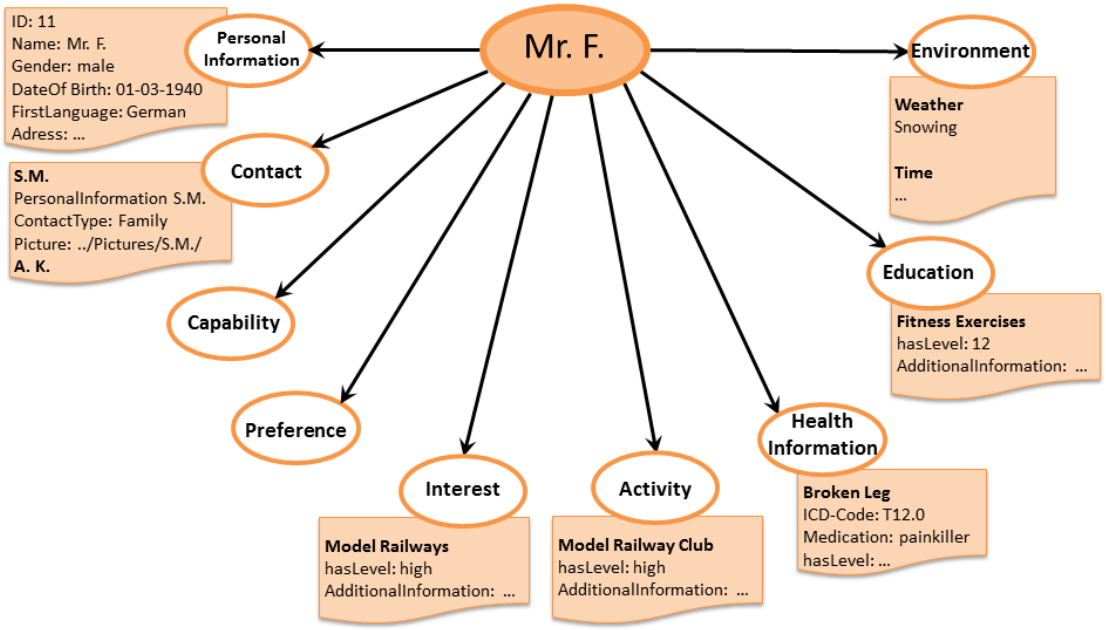


Figure 7: PCEICL ontology example

in the class *SystemPreferences*. Other information must be collected manually, like information about existing accounts.

For the organization of club meetings, information about the other club members is needed. This is also realized by the *Contact* class, which offers the possibility to order the contacts in a category *club members*, so the app can show only the relevant club persons. When the elderly person wants to organize a meeting at his home or when the system searches for a ride to an external meeting or event, this grouping functionality gives great support.

B. Use Case (Information Acquisition Assistance):

For the application which assists the user in organizing information acquisition, primarily knowledge about the health condition (ontology class: *HealthCondition*) and environment (ontology class: *Environment*) of the user is helpful. The main advantage of this app is, that it asks if it should organize assistance in something depending on the actual context. Context could be for example the condition of the user, his impairments, the weather, the time, the sensory in the house, etc. Due to the information given by the concepts of the ontology, the system can decide if there is a need for a special service or not. The user is always asked if help by the system is needed without being patronizing.

Due to the acquisition of the interests, activities and personal preferences of the user, it is possible to offer the user the "search-and-offer" service assistance of finding all kinds of support. Since the user in our use cases has a broken leg, he could get a snow shoveling offer automatically, during winter time. The model railway passion of Mr. F. can be supported by helping him to get a lift for the weekly meetings by the "search-and-offer" service. It uses the information saved in the instances of the classes *Interest* and *Activity* for providing personalized functions.

C. Use Case (Learning Assistance):

The fitness status of Mr. F. can be gathered by the *Education* class. If he reaches a new level of fitness by doing all the required exercises, the level can be saved in this class. Through the *History* class, it is possible to consider the whole progress of fitness condition. If there are steps backwards, the exercises could be adapted accordingly. The health condition information makes it possible to automatically offer only the exercises that are feasible for the user with his current impairment.

VII. PCEICL Ontology Summary

There are several advantages of the newly introduced PCEICL ontology: (1) centralized user view, (2) services adaptable to user's needs, (3) historical view, (4) usage of the ontology for ACL, (5) expandability. These advantages may be summarised as follows:

(1) Context awareness is a mandatory requirement for an optimal assistance for elderly people. It is also important not to consider only the environment of assisted persons, but especially the assisted persons themselves and their needs. The PCEICL ontology offers a **centralized user view**. Therefore, useful information about the user can be applied to offer personalized services.

(2) All **services are adaptable to the user's needs**. They can use information about the capabilities, impairments, interests, etc. of the person and can tailor their user interface and also their functionality to the abilities of the user. For example, they can control the volume, the font size or the colours, depending on the condition of the ears or the eyes of the user.

(3) Due to the **historical view** integrated into the ontology, it is also possible to react to changes of the user profile. The services can, for example, repeat helpful information if the user begins to suffer from dementia.

 ISBN: 978-1-61208-337-7

(4) The ontology is used within the PCEICL platform as a common understanding of the user profile data. It is used to semantically interpret the user context in the different use cases and during the **communication between the agents** of the PCEICL platform.

(5) Another benefit of the ontology is its facilitated **expandability**. This is because of the centralized concept *User*, where it is easy to add new properties to expand the user profile. Other reasons are the hierarchical structure and the reuse of concepts by many classes. Only currently useful information is saved in the ontology for a better data privacy. Concepts, which turn out to be relevant in the future, can be easily integrated into the ontology.

VIII. Conclusion and future work

In this paper, we presented an ontology, which constitutes the base of a context aware platform as a service. The PCEICL platform itself is based on OSGi with the opinion to use JADE agents for the integration of semantic intelligence in form of an ontology. The presented PCEICL ontology is the base for ACL communication of the agents. In the future, there will be an OSGi store, where the OSGi bundles can be distributed. Due to the use of cloud technologies, the platform is flexible and scales as needed.

The use of ontologies in the field of AAL is not new, but the centralized role of the user in context modelling is not widespread. In the PCEICL ontology, the user is the central concept. The user is described by his properties like health condition, capabilities, preferences, his social environment, etc. For the exact and correct description of the user's health condition, the ICD-Code *ICD-Code* is used, which is a worldwide applied classification system for diagnoses. With the concepts of the ontology it is also possible to have a historical view on the user and his environment. So, it is possible to analyse the user's development of learning or the development of the health condition to permanently adapt the system to the needs of the user. Due to the centralized user view, also a better expandability of the ontology is reached.

Future work will be a dynamic adaption of the ontology during runtime. Some context information will be captured and analysed automatically and the result could lead to a modification of the ontology. For example, the system could monitor the behaviour and the search requests of the user and could conclude that the interests of the user changed. In this case an adaption of the interest instance in the ontology should occur. Some other information shouldn't be captured automatically. Such an information is, for example, the healing of a disease and should be detected by a doctor. After this detection, the ontology should adapt accordingly.

Acknowledgment

The project ZAFH-AAL ("Zentrum für Angewandte Forschung an Hochschulen für Ambient Assisted Living") is funded by the Ministry of Science, Research and the Arts of Baden-Württemberg, Germany. The funding program for the universities of applied science is called: Zukunftsoffensive IV "Innovation und Exzellenz" (ZO IV). The PCEICL project is a sub-project of the project ZAFH-AAL.

References

[1] M. Klein, A. Schmidt, and R. Lauer, "Ontology-Centred Design of an Ambient Middleware for Assisted Living: The Case of SOPRANO," in In: Towards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments (AIM-CU), 30th Annual German Conference on Artificial Intelligence (KI 2007), 2007.

[2] L. Litz and M. Gross, "Covering Assisted Living Key Areas based on Home Automation Sensors," in Networking, Sensing and Control, 2007 IEEE International Conference on, 2007, pp. 639–643.

[3] J. A. Botia, A. Villa, and J. Palma, "Ambient Assisted Living system for in-home monitoring of healthy independent elders," Expert Systems with Applications, vol. 39, no. 9, 2012, pp. 8136 – 8148.

[4] S. Lauriks et al., "Review of ICT-Based Services for Identified Unmet Needs in People with Dementia," in Supporting People with Dementia Using Pervasive Health Technologies, ser. Advanced Information and Knowledge Processing. Springer London, 2010, pp. 37–61.

[5] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Springer-Verlag, 1999, pp. 304–307.

[6] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," in Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications, ser. WMCSA '94. Washington, DC, USA: IEEE Computer Society, 1994, pp. 85–90.

[7] M. Assad, D. Carmichael, J. Kay, and B. Kummerfeld, "PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services," in Pervasive Computing, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4480, pp. 55–72.

[8] R. Ram et al., "universAAL: Provisioning Platform for AAL Services," in Ambient Intelligence - Software and Applications, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2013, vol. 219, pp. 105–112.

[9] "OSGi Alliance," [retrieved: March, 2014]. [Online]. Available: http://www.osgi.org/

[10] K. Skillen, L. Chen, C. Nugent, M. Donnelly, and I. Solheim, "A user profile ontology based approach for assisting people with dementia in mobile environments," in Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, 2012, pp. 6390–6393.

[11] K.-L. Skillen et al., "Ontological User Profile Modeling for Context-Aware Application Personalization," in Ubiquitous Computing and Ambient Intelligence, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7656, pp. 261–268.

[12] P. Moreno, M. Hernando, and E. Gómez, "AALUMO: A User Model Ontology for Ambient Assisted Living Services Supported in Next-Generation Networks," in XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013, ser. IFMBE Proceedings, L. M. Roa Romero, Ed. Springer International Publishing, 2014, vol. 41, pp. 1217–1220.

[13] D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. Wilamowitz-Moellendorff, "Gumo - The General User Model Ontology," in User Modeling 2005, ser. Lecture Notes in Computer Science, L. Ardissono, P. Brna, and A. Mitrovic, Eds. Springer Berlin Heidelberg, 2005, vol. 3538, pp. 428–432.

[14] M. Sutterer, O. Droegehorn, and K. David, "UPOS: User Profile Ontology with Situation-Dependent Preferences Support," in Advances in Computer-Human Interaction, 2008 First International Conference on, 2008, pp. 230–235.

[15] "eHealth / AAL," [retrieved: March, 2014]. [Online]. Available: http://www.prosyst.com/what-we-do/ehealth-aal/products/

[16] "Agent Communication Language Specifications," [retrieved: March, 2014]. [Online]. Available: http://www.fipa.org/repository/aclspecs.html

[17] "Java Agent DEvelopment Framework," [retrieved: March, 2014]. [Online]. Available: http://jade.tilab.com/

[18] "The Foundation of Intelligent Physical Agents," [retrieved: March, 2014]. [Online]. Available: http://www.fipa.org/

[19] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Online, 2001, [retrieved:

 ISBN: 978-1-61208-337-7

March, 2014]. [Online]. Available: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html

[20] C. Fredrich, “Kontextgetriebene "Software as a Service" im Bereich "Ambient Assisted Living",” Master’s thesis, Hochschule Furtwangen University, 2013, [retrieved: March, 2014]. [Online]. Available: http://www.wolke.hs-furtwangen.de/assets/files/Theses/2013_SS13_Carina_Fredrich_Kontextgetriebene_SaaS_im_Bereich_AAL.pdf

[21] “World Health Organization (WHO): International Classification of Diseases (ICD),” [retrieved: March, 2014]. [Online]. Available: http://www.who.int/classifications/icd/en/

 ISBN: 978-1-61208-337-7

Analyzing Behavioral Compatibility for Web Service Choreography Using Colored Petri Nets and ASK-CTL

Maya Souilah Benabdelhafid
Constantine2 University
LIRE Laboratory
Chaab Essas
Constantine, Algeria
Email: mabenabdelhafid@gmail.com

Béatrice Bérard
Sorbonne University
LIP6 Laboratory
UPMC & CNRS
Paris, France
Email: beatrice.berard@lip6.fr

Mahmoud Boufaida
Constantine2 University
LIRE Laboratory
Chaab Essas
Constantine, Algeria
Email: mboufaida@umc.edu.dz

***Abstract*—Web services have become the technology of choice for Service-Oriented Computing (SOC) implementation. Their composition is a recent field that has seen a flurry of different approaches proposed towards the goal of flexible distributed heterogeneous inter-operation of software systems. These systems are usually derived from higher-level models rather than be coded at low level. In practice, achieving Web service compatibility nonetheless continues to require significant efforts for modeling at multiple abstraction levels. Existing formal approaches typically require the analysis of the global space of joint executions of interacting Web services. We propose a formal approach where Web service choreography is represented with the high-level model of Colored Petri Nets (CPNs). ASK-Computational Tree Logic (ASK-CTL) is used to describe the behavioral compatibility of these services in terms of message order properties. Then, model checking is applied for the verification of these properties. The effectiveness of our work has been validated with the recent version of CPN Tools.**

***Keywords*-Web Service Choreography; Behavioral Compatibility; Model Checking; CPN; ASK-CTL.**

I. INTRODUCTION

SOC is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments [1]. Web services [2] are considered as one of the most promising computing paradigms, which work as plugin mode to provide the value-added applications in SOC and Service-Oriented Architecture (SOA) [3]. They may use the Internet as the communication medium and open Internet-based standards, such as the Simple Object Access Protocol (SOAP) as transmission medium and the Web Services Description Language (WSDL) for their description. They currently support the externalization of atomic business capabilities [4]. Specifically, it is commonly accepted that a Web service description should include not only the interface, but also the business protocol supported by the service (i.e., its behavior, which is the specification of possible message exchange sequences that it supports). Services can be composed through choreography and orchestration. Choreography describes the interactions between participating services to the business process from a global perspective, while orchestration uses a central coordinator. Many composition methods as well as several proposals, such as Web Services Business Process Execution Language (WSBPEL) [2] for orchestration or Web Service Choreography Definition Language (WSCDL) [5] for choreography, have been brought forward to construct and describe the interactions among services. However, they are concerned only with syntactic or semantic compatibility among services, and the behavioral compatibility is ignored.

Behavioral compatibility analysis for Web service composition is one of the most important topics. In this paper, our goal is to investigate this topic in the context of choreography. We provide a formal basis for developing demonstrably correct choreography. Our definition for this correctness is related to message order requirements. We consider the problem of choreographing Web services from a high-level, conceptual perspective, that abstracts from the details of the interaction paradigm. As pointed by De Backer et al. [6], the first step of verifying if two Web services are compatible should occur on an abstract level that hides unnecessary underlying coordination and allows to focus on high-level units of collaboration. This simplifies the verification and provides a first step towards a compatibility before investigating details of a Web service description such as the content of a message. We propose the modeling of Web services and their choreography using CPNs [7] and show how a model checking technique can be employed to verify if the modeled choreography satisfies the order properties given as ASK-CTL [8] formulas. The CPN models are implemented using the recent version of the software CPN Tools (CPN Tools 4.0 [9]). The ASK-CTL toolkit provided with this tool is used to perform automated verification in order to prove that a service choreography is correct at design time. This is an important step towards reliable service choreography composition, since problems could be detected early in the development cycle, before even starting the implementation.

The rest of this paper is structured as follows. In Section II, we give a simple illustrating example of Web service behaviors in a choreography. Formal definitions of CPNs and ASK-CTL are recalled in Section III, with a brief description of the model checking technique. The formalization of behavioral

 ISBN: 978-1-61208-337-7

compatibility is presented in Section IV. Related works are discussed in Section V. Conclusions and future works are presented in Section VI.

II. Motivating Example

Let us consider a simple example where the scenario is that of a travel agency, with the cooperation of four partners:

1) *Travel Agency* has two main tasks: airline booking and hotel reservations,
2) *Bank* acts as a financial intermediary between the Airline company (respectively the Hotel) and the Travel Agency,
3) *Airline Company* sells flight tickets to Travel Agencies,
4) *Hotel* proposes nights to Travel Agencies.

The last three partners want to provide functionalities to the Travel Agency partner using the Web service technology. Each partner is a published Web service, participating in a choreography and is modeled as a business process including the description of its partners (or a link permitting to get it), the description of its interface (but not its local operations), and the description of an abstract process that represents its behavior (exchanged messages). The behavior of the four Web services is as follows (see Fig. 1.).

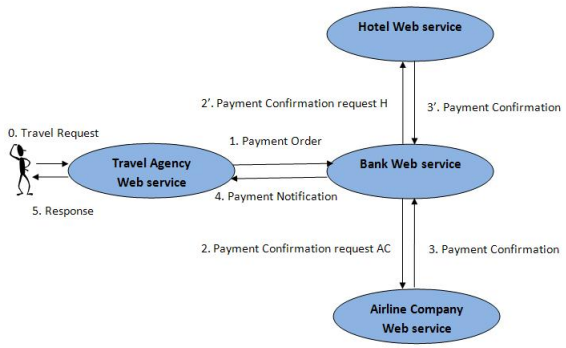


Fig. 1. A Web service choreography: A Travel Agency Example

First, a customer contacts the Travel Agency Web service and chooses its travel plan including information about the order and the payment method. Consequently, this service contacts the Bank Web service to pay the Airline Company (respectively the Hotel) Web service. Next, the Bank pays the Airline Company (respectively the Hotel) and asks them for the payment confirmation. The Airline Company (respectively the Hotel) sends its confirmation. If the payment operations are completed successfully then the Travel Agency contacts its customer and confirms his travel plan, and if one of them fails then it contacts the customer and asks if any other plan suits him or to cancel his request. The messages exchanged between the four Web services have constraints of order forming their behaviors.

The possible scenarios can be the following message ordering sequences: $0, 1, 2, 3, 2', 3', 4, 5$ or $0, 1, 2, 2', 3, 3', 4, 5$ or $0, 1, 2, 2', 3', 3, 4, 5$ or $0, 1, 2', 3', 2, 3, 4, 5$ or $0, 1, 2', 2, 3', 3, 4, 5$ or $0, 1, 2', 2, 3, 3', 4, 5$.

To guarantee the successful execution of these scenarios, Web services need to be verified formally in order to ensure that mutual interactions between them do not lead to any conflict. Specifically, we need to verify their compatibility. There are three aspects of service compatibility: syntactic, semantic, and behavioral [10]. Syntactic compatibility means that the structural interfaces of the interacting services are consistent. Semantic compatibility means that the interacting services exchange information that can be understood in a consistent and unambiguous way. Finally, behavioral compatibility means that the interacting services agree on what to expect from each other in terms of operations to execute, outcomes to deliver, and messages to be sent and received.

The static compatibility including the syntactic and semantic compatibility is essential to be checked. Checking the behavioral one, however, is a much more challenging task. In the example, the four partners may be syntactically and semantically compatible in interfaces, but they can behave improperly for the message exchange protocol. An example of behavioral property that we will later check is the following requirement: *The payment confirmation will be sent by the Airline Company after it receives the payment confirmation request.* It is obvious that if this property is not satisfied, then the collaboration leads to an erroneous message ordering even if they are syntactically and semantically consistent. Thus, the behavior of services must be taken into account in composition. The manual checking of service compatibility would clearly be error-prone and time consuming. Consequently, an approach to realize automatic and transparent checking is necessary.

This example will be modeled and the above behavioral property will be verified in order to respect the six anticipated scenarios.

III. Background

In this section, we briefly recall some formal definitions related to model checking formulas of the ASK-CTL logic for CPNs.

A. Colored Petri Nets

CPNs represent today one of the most widely used formalism incorporating data and hierarchy [11]. They are a discrete-event modeling language combining PNs and the functional programming language CPN ML. Initially, CPNs were supported by Design/CPN, later replaced by CPN Tools that supports the design of complex processes and the analysis of such processes using simulation and state space analysis.

In this section, we first recall definitions of CPNs that will be useful in establishing a CPN model for Web service choreography. These definitions are presented here in a simple way in order to adapt them to our problem of behavioral compatibility. A relation between them and CPN Tools 4.0 notations is also presented.

Definition 1 (Multi-set)**.** *A multi-set over a non-empty set Z is a mapping $b : Z \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. The support of b is the set $supp(b) = \{z \in Z \mid b(z) \neq 0\}$. We denote by $Bag(Z)$ the set of multi-sets over Z with*

finite support and we write sometimes explicitly $b \in Bag(Z)$ *as* $b = \sum_{z \in Z} b(z)z$. *An* order relation, *an* addition *and a* difference *on multi-sets are defined as follows. For two multi-sets* $b, b' \in Bag(Z)$*:*

- $b \leq b'$ *if for all* $z \in Z$, $b(z) \leq b'(z)$,
- $b + b'$ *is given by* $\sum_{z \in Z}(b(z) + b'(z))z$.
- *if* $b \leq b'$, *then* $b' - b = \sum_{z \in Z}(b'(z) - b(z))z$.

Remark 1. In CPN Tools 4.0, special symbols are used for multi-sets: the order relation is noted $\ll$, the addition is noted $++$, with $\overset{++}{_{MS}\sum}$ for a sum, and the symbol ‘ is placed between $b(z)$ (the multiplicity) and z (the element). These notations appear in Figures 3 and 4 for instance, where CPNs are extracted from the graphical interface of the tool.

CPN definitions use a set Σ of color domains containing the set $Bool = \{true, false\}$ and a set V of variables. The variables are typed by the function $Type : V \rightarrow \Sigma$ and we consider a set $Exp(V)$ of expressions using elements of V as free variables (or the empty expression).

Definition 2 (CPN Syntax)**.** *A CPN over the set of color domains* Σ *and the set of variables* V *is a 5-uplet* $N = (P, T, C, E, M_0)$ *where:*

- P *is a finite set of* places,
- T *is a finite set of* transitions *such that* $P \cap T = \emptyset$,
- $C : P \rightarrow \Sigma$ *associates a* color domain *with each place,*
- $E : P \times T \cup T \times P \rightarrow Exp(V)$ *associates with each pair* (p, t) *or* (t, p) *an* expression *typed as a multi-set over the color domain of the place:* $Type(E(p, t)) = Bag(C(p))$ *and* $Type(E(t, p)) = Bag(C(p))$,
- M_0 *is the* initial marking, *with* $M_0(p) \in Bag(C(p))$ *for each place* $p \in P$.

Remark 2. We do not define explicitly the set of arcs to simplify the notations and we use the habitual convention of Petri nets: the expression is empty if there is no arc, an empty expression evaluating to an empty multi-set.

The following definition presents the semantics of CPNs.

Definition 3 (CPN semantics)**.** *The semantics of a CPN* N *is described by a transition system* $\mathcal{T}_N = (\mathcal{M}, M_0, \rightarrow)$*:*

- *the configurations of* $\mathcal{M}$ *are markings* M, *with* $M(p) \in Bag(C(p))$ *for each place* $p \in P$,
- *the initial configuration is the initial marking* M_0,
- *the* transition relation $\rightarrow$ *is defined as follows.*
  Let t *be a transition and let* v *be a valuation of variables. We write* $v^-(p, t) \in Bag(C(p))$ *and* $v^+(t, p) \in Bag(C(p))$ *for the respective values of* $E(p, t)$ *and* $E(t, p)$ *for* $p \in P$.
  The transition $M \xrightarrow{t,v} M'$ *is possible if, for each place* $p \in P$, $M(p) \geq v^-(p, t)$ *and in this case,*
  $M'(p) = M(p) - v^-(p, t) + v^+(t, p)$ *for each* $p \in P$.

An execution *starting from* M *is a sequence of firings* $M \xrightarrow{t_1,v_1} M_1 \xrightarrow{t_2,v_2} M_2 \ldots$. *A marking* M' *is reachable from* M *if there exists a finite execution* $M \xrightarrow{t_1,v_1} M_1 \xrightarrow{t_2,v_2} M_2 \ldots \xrightarrow{t_n,v_n} M_n$ *starting from* M *such that* $M' = M_n$.

Remark 3. In this definition, a single transition is fired to avoid the steps in the presentation of Jensen [7]. This is not a problem because a step can be represented by the successive firing of several transitions.

B. The logic ASK-CTL

The logic ASK-CTL of CPN Tools (see [12] for more details) is an extension of the standard CTL [13]. An ASK-CTL formula is interpreted over the transition system $\mathcal{T}_N$ (called State Space (SS) in the tool) associated with a CPN model N and takes into account both configuration information (on markings, also called states) and transition information, thus extending CTL, where only configurations are labeled with sets of atomic propositions. The model checker of CPN Tools checks if such a formula holds over $\mathcal{T}_N$.

In the following definition, we consider the transition system $\mathcal{T}_N$ of a given CPN N. Operators $\neg, \wedge$ are boolean negation and conjunction, $\langle . \rangle$ is an existential ”next” modality, U is the standard until modality of CTL and E, A are respectively the existential and universal quantifiers on executions from CTL.

Definition 4 (ASK-CTL Syntax)**.** *The ASK-CTL logic has two categories of formulas: state and transition formulas, defined by mutual induction.*

State formulas are given by the grammar:
$\mathcal{A} ::= \alpha | \ \neg\mathcal{A} \ | \mathcal{A}_1 \wedge \mathcal{A}_2 | \ \mathsf{EU}(\mathcal{A}_1, \mathcal{A}_2) | \ \mathsf{AU}(\mathcal{A}_1, \mathcal{A}_2) | \ \langle\mathcal{B}\rangle$
where α *is a mapping from the set* $\mathcal{M}$ *of markings into booleans,* $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$ *are state formulas and* $\mathcal{B}$ *is a transition formula.*

Transition formulas are given by the grammar:
$\mathcal{B} ::= \beta | \ \neg\beta \ | \beta_1 \wedge \beta_2 | \ \mathsf{EU}(\beta_1, \beta_2) | \ \mathsf{AU}(\beta_1, \beta_2) | \ \langle\mathcal{A}\rangle$
where β, β_1, β_2 *are mappings from the set of pairs* (t, v) *labelling transitions into booleans and* $\mathcal{A}$ *is a state formula.*

The semantics of ASK-CTL is defined inductively on configurations of the transition system $\mathcal{T}_N$ in the spirit of CTL, from the basis case: A configuration M satisfies α, written $M \models \alpha$, if $\alpha(M)$ is true. For instance:
- $M \models \mathsf{EU}(\mathcal{A}_1, \mathcal{A}_2)$ if there exists an execution $M \xrightarrow{t_1,v_1} M_1 \xrightarrow{t_2,v_2} M_2 \ldots \xrightarrow{t_n,v_n} M_n$ starting from M such that M_n satisfies $\mathcal{A}_2$ and all markings from M to M_{n-1} satisfy $\mathcal{A}_1$.
- $M \models \mathsf{AU}(\mathcal{A}_1, \mathcal{A}_2)$ if for all executions starting from M, there exists a marking M' satisfying $\mathcal{A}_2$ with all intermediate markings satisfying $\mathcal{A}_1$.

The next modality is similar to the one from the μ-calculus: $M \models \langle\mathcal{B}\rangle$ if there is a transition $M \xrightarrow{t,v} M'$ from M satisfying $\mathcal{B}$, as defined below.

The semantics of transition formulas is defined similarly:
- A transition $e = M \xrightarrow{t,v} M'$ satisfies β if $\beta(t, v)$ is true.
- $e \models \langle\mathcal{A}\rangle$ if M' satisfies $\mathcal{A}$.
- The formulas $\mathsf{EU}(\beta_1, \beta_2)$ and $\mathsf{AU}(\beta_1, \beta_2)$ are then defined like above on executions starting by e, with β_1 and β_2 satisfied by successive transitions instead of configurations.

Note that we may also use the standard abbreviations $\mathit{false} = \alpha \wedge \neg\alpha$, $\mathit{true} = \neg\mathit{false}$, $\varphi \rightarrow \psi = \psi \vee \neg\varphi$, $\mathsf{AF}\varphi = \mathsf{AU}(true, \varphi)$ and $\mathsf{AG}\varphi = \neg\mathsf{AF}(\neg\varphi)$.

 ISBN: 978-1-61208-337-7

C. Model Checking

A model checking procedure answers the following question: Given a state ASK-CTL formula $\mathcal{A}$ and a CPN N, does the initial configuration M_0 of $\mathcal{T}_N$ satisfy $\mathcal{A}$? For this, it must be able to answer any similar question on reachable configurations or transitions of $\mathcal{T}_N$. The tool uses Standard ML (SML) functions for this purpose. For instance, checking a state formula is expressed in SML by a function $eval\_node :< formula >, < node >$, which takes two arguments: the formula to be checked and a configuration (called node in the tool) from where the model checking should start. The mappings α are defined in SML by functions like $\mathsf{NF}(< message >, < node\ function >)$, where $node\ function$ takes a node and returns a boolean and $message$ is used when a formula evaluates to false. Similarly, the mappings β are defined by functions like $\mathsf{AF}(< message >, < arc\ function >)$. A formula $\mathsf{EU}(\alpha_1, \alpha_2)$ for two mappings α_1 and α_2, simply translates in SML as $\mathsf{EXIST\_UNTIL}(\alpha_1, \alpha_2)$, and so on.

Now, we deal with our proposed formal approach.

IV. A CPN AND ASK-CTL -BASED APPROACH

Our approach analyzes behavioral compatibility using the above definitions of CPNs and ASK-CTL. It is composed of two related phases (see Fig. 2.):

- *Choreography Modeling and Validation*: Modeling a Web service choreography by constructing Web service behaviors based on CPNs semantics and composing them. This modeling is validated by multiple simulations using CPN Tools 4.0. The result is a behavioral model to check.
- *Behavioral Properties Checking*: Verifying some behavioral properties on the generated behavioral model in terms of message order using the model checking technique described above. We first formally describe the behavioral properties as ASK-CTL formulas. Subsequently, we rewrite these formulas into SML format. In this way, a concrete formalization of the behavioral properties is obtained. The verification of these properties will be done over the transition system (or *SS*) that has been generated from the behavioral model by CPN Tools 4.0.

A. Choreography Modeling and Validation

In this first phase, we have three related steps: *CPN Modeling, Simulation, and CPNs Composition*.

1) CPN Modeling: According to the Web service behaviors, which are specified by the informal language Unified Modeling Language Diagram Activities (UML DA [14]), we construct a formal model for each Web service behavior based on CPN semantics such as the choreography may require different instances of a participating Web service. Consequently:

- the behavior execution states are captured by places.
- the message type (Web service instances and its incoming messages) is captured by the color set of the token (we do not look into the content of a message as it is not known until run time).
- the operation of its instance is captured by a transition (send or receive).

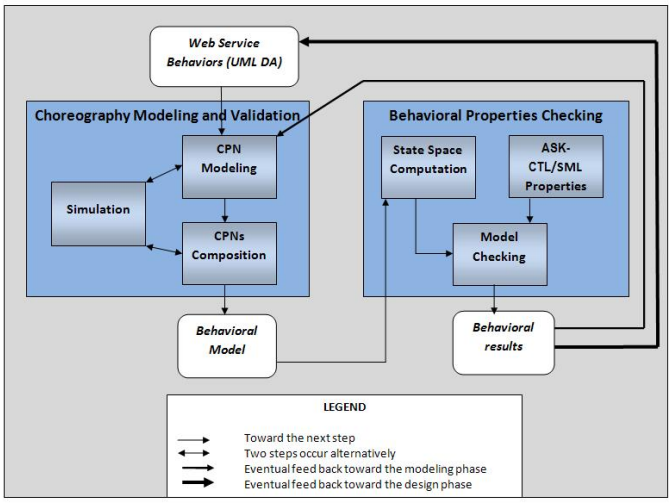


Fig. 2. Overview of our proposed approach

- the Web service initial state is captured by the initial marking M_0.

In our modeling, a Web service behavior is a conversation protocol that is defined as a CPN N where:

- the set of colors is $\Sigma = \{INS, I, MSGSTATE, I \times MSGSTATE, INS \times I \times MSGSTATE\}$, where:
  - INS is a color set, which defines the Web service instances:
    $colset\ INS = with\ ins1|ins2;$
    We define a variable x having as type INS:
    $var\ x : INS;$
  - the static subclasses of $I \times MSGSTATE$ include I, which is an integer type that represents the message identifier and $MSGSTATE$, which is an enumeration type that represents a message:
    $colset\ I = int;$
    $colset\ MSGSTATE = with\ TravelRequest$
    $|Response|PaymentOrder|PaymentNotification$
    $|PaymentConfirmationRequestAC$
    $|PaymentConfirmationRequestH$
    $|PaymentConfirmation;$
    we define two variables $msgid$ and $currentstate$ and $m0, m1, m4, m5$ having respectively as type I, $MSGSTATE$, and $I \times MSGSTATE$:
    $var\ msgid : I;$
    $var\ currentstate : MSGSTATE;$
    $var\ m0, m1, m4, m5 : I \times MSGSTATE;$
  - we define two variables $xm2, xm3$ having as type $INS \times I \times MSGSTATE$:
    $var\ xm2, xm3 : INS \times I \times MSGSTATE;$
  - we define also functions that will be attached to transitions and eventually two arcs, for example the function $startSend1$ that allows the sending of the message $m1$ that represents the Payment Order in Fig. 1. It is defined as follows:
    $fun\ \ startSend1((msgid, currentstate)\ :\ I\ \times$

$MSGSTATE) = let\ val\ new\_msgid = 1$
$val\ new\_currentstate = PaymentOrder$
$in\ (new\_msgid, new\_currentstate)\ end$

- a place $p \in P$ represents the protocol state, a transition $t \in T$ represents the message exchange consisting on an invocation of a Web service operation, and the initial marking M_0 represents the Web service initial state.

For instance a part of the Travel Agency Web service behavior is shown in Fig. 3.

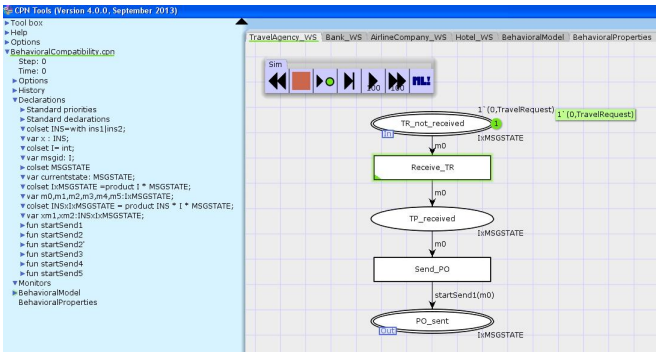


Fig. 3. A CPN modeling of a part of the Travel Agency Web service behavior

As we can see, each of the Travel Agency Web service operation is represented by a transition. The initial marking consists in the token in the place $TR\_not\_received$. The relations between operations are modeled by the firing rules of the CPN:

- $Receive\_TR$ is the first transition that can be fired if the token $(0, TravelRequest)$ is present in its input place.
- $Send\_PO$ will then be fired with the function $startSend1$ defined above.

We can now describe the CPN that models this part of the Travel Agency Web service behavior.

1) $\Sigma = \{INS, I, MSGSTATE, I \times MSGSTATE, INS \times I \times MSGSTATE\}$,
2) $P = \{TR\_not\_received, TP\_received, PO\_sent\}$
3) $T = \{Receive\_TR, Send\_PO\}$
4) $E(TR\_not\_received, Receive\_TR) = E(Receive\_TR, TP\_received) = E(TP\_received, Send\_PO) = m0, \ldots$
5) $M_0 = \{(0, TravelRequest)\ ++\ ins1\}$.

A finite execution of the protocol of the Travel agency Web service is defined by a sequence $M_0 \xrightarrow{Receive\_TR, msgid=0, currenstate=TravelRequest} M_1 \xrightarrow{Send\_PO, msgid=1, currenstate=PaymentOrder} M_2, \ldots$

2) Simulation: The above figure showed that the marking M_0 of the Travel Agency model changed to another marking M_1 after the occurrence of the transition *Receive_TR*. Another transition could be enabled and fire as a result of the new marking. This process of firing of a sequence of transitions is called *simulation*. Fig. 3. shows the simulation tool palette used for validating the Travel Agency Web service behavior. We note that simulations are also performed on CPNs composition step. In addition, simulations analyze a finite number of executions and help to validate the model by detecting and finding errors in the CPN model and demonstrates that the model works correctly. However, it is impossible to guarantee the correctness of the model with 100% certainly because all the possible executions are not covered [15]. This correctness will be analyzed in the second phase of our approach.

3) CPNs Composition: From the Web service behaviors that have been modeled on CPN models, we can now perform their composition using the concept of *sub-module* and the result will be a formal model that represents the Web service choreography called *behavioral model*. In this composition, the CPN models can be structured into a set of sub-modules to handle large specifications. These modules-pages interact with each other through a set of well-defined interfaces, in a similar way to programming languages. Fig. 4. shows the CPNs composition where we have four sub-modules (T, B, H, AC) representing respectively the four Web services behaviors (Travel Agency, Bank, Hotel, and Airline Company).

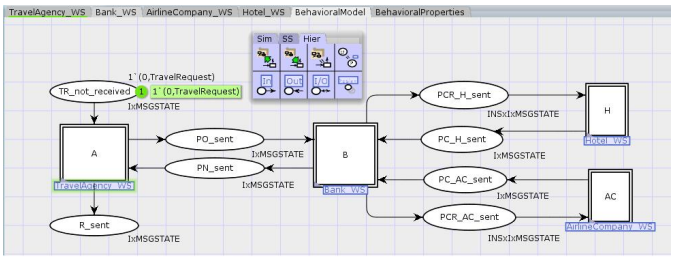


Fig. 4. CPNs Composition

For example, the two places *TR_not_received* and *PN_sent* represent *input ports* for the T sub-module. The two places *PO_sent* and *R_sent* are its *output ports*. This means that these places form the interface through which the T sub-module exchanges tokens with the other sub-pages. It will import tokens via the input ports and it will export tokens via the output ports. The composition of p sub-modules $N_1, \ldots N_p$ is denoted by $N_1 \oplus \cdots \oplus N_p$. Since we have composed our CPNs models representing the taken Web services, we can substitute each sub-module by its corresponding CPN. As said above, simulation is performed on the composition to validate it but, it is not sufficient to prove its behavioral compatibility. To do so, we perform the next phase.

B. Behavioral Properties Checking

From a generated behavioral model representing the Web service choreography, the behavioral properties checking can be performed through the verification of the message order properties. The notion of syntactical and semantic compatibility are preconditions of the following checking. Also, we consider the case where the component Web services in a choreography have correct behaviors. In this case, whether the composition can properly execute or not depends on the behavioral compatibility of its participating Web services.

Definition 5 (Behavioral Compatibility)**.** *Let $N = N_1 \oplus N_p$ be a CPN representing the behavioral model produced by*

the composition of p CPNs $N_1, \ldots, N_p$ representing the Web service models. Let $(i, j, request, m)$ denote transition labels for the sending of a request m from service i to service j and let $(i, j, answer, m)$ denote the transition label for the answer to this request from j to i. Then, N is behaviorally compatible with respect to message ordering if for all i, j, m, the following state formulas are satisfied by the initial configuration of N :

- $\mathsf{AG}(\langle(i, j, request, m)\rangle \rightarrow \mathsf{AF}(\langle(i, j, answer, m)\rangle))$, *meaning that any request is eventually followed by an answer, and*
- $\mathsf{AU}(\neg\langle(i, j, answer, m)\rangle, \langle(i, j, request, m)\rangle)$, *meaning that no answer is sent until a request has been sent first.*

Justification: We recall that each Web service is represented by a CPN, each Web service interaction (send or receive) is represented by a transition, and each exchanged message is represented by a color set of the token. Analyzing the behavioral compatibility of a Web service choreography is subject to verifying its correctness. This correctness is related to some qualitative requirements that are set on the order of the exchanged messages. We note that the second formula corresponds to the property given in Section II as example for the case study: *The payment confirmation will be sent by the Airline Company after it receives the payment confirmation request*. Both of the two formulas will be verified using a model checking technique based on SS. Thus, in this second phase, we have three related steps: *State Space Computation, ASK-CTL/SML Property Description, and Model Checking*.

1) State Space Computation: Our approach verifies the behavioral compatibility of a Web service choreography by using CPN Tools to automatically generate the transition system $\mathcal{T}_N$ associated with the choreography model. Only nodes reachable from the initial marking M_0 of the net and the associated transitions are kept by the tool. For our example above, the transition system $\mathcal{T}_N$ has 17 nodes (see Fig. 5.) representing the different markings, generated by all transitions:
$M_0 \xrightarrow{x=ins1,Receive\_TR,msgid=0,currentstate=TravelRequest} M_1$
$M_1 \xrightarrow{x=ins1,Send\_PO,msgid=1,currentstate=PaymentOrder} M_2$
and so on, up to
$M_{16} \xrightarrow{x=ins1,Send\_Response,msgid=5,currentstate=Response} M_{17}$

The transition system $\mathcal{T}_N$ can be used not only to obtain a standard report (including standard properties such as deadlock freeness) but also to verify ASK-CTL formulas like those defined for compatibility.

2) ASK-CTL/SML Properties Description: ASK-CTL formulas are used here to describe the behavioral properties to be checked. Let us deal with the behavioral property taken for our example (corresponding to a formula of the second type in Definition 5).

Behavioral Property: *The payment confirmation will be sent by the Airline Company after it receives the payment confirmation request.*

We rewrite the corresponding ASK-CTL formula into SML to obtain a concrete formalization of the property (see Table I). This formula is given by $\mathsf{AU}(\neg A_2, A_1)$ where A_1 denotes the characteristic predicate for the transition of receiving the payment confirmation request by the Airline Company Web service and A_2 denotes the characteristic predicate for the transition of sending the payment confirmation by the same Web service.

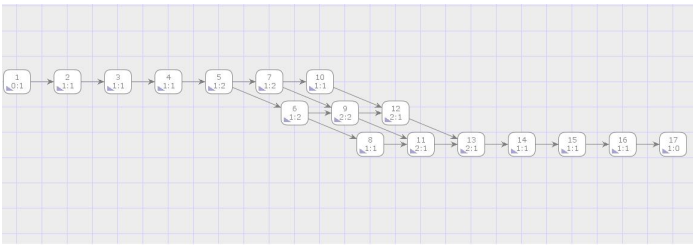


Fig. 5. Transition system of our behavioral model

TABLE I
SML FUNCTIONS FOR CHECKING THE BEHAVIORAL PROPERTY OF THE EXAMPLE

	SML Description
Functions and values declaration	$fun\ Arc1$ $a = (Bind.BehavioralModel'Receive\_PCR\_AC$ $(1, \{xm2 =$ $(ins1, 2, PaymentConfirmationRequestAC)\})$ $= ArcToBE\ a);$ $fun\ Arc2$ $a = (Bind.BehavioralModel'Send\_PC\_AC$ $(1, \{xm2 =$ $(ins1, 2, PaymentConfirmationRequestAC)\})$ $= ArcToBE\ a);$ $valA1 = AF("Receive", Arc1);$ $valA2 = AF("Send", Arc2);$
Formula	$val\ myASKCTLformula =$ $FORALL\_UNTIL(NOT(A2), A1);$
Verification	$eval\_arc\ myASKCTLformula\ InitNode;$

In this description, A_1 is interpreted by:
$fun\ Arc1$
$a = (Bind.BehavioralModel'Receive\_PCR\_AC$
$(1, \{xm2 = (ins1, 2, PaymentConfirmationRequestAC)\});$
referring the variable $xm2$ of transition $Receive\_PCR\_AC$.

And A_2 is interpreted by:
$fun\ Arc2$
$a = (Bind.BehavioralModel'Send\_PC\_AC$
$(1, \{xm2 = (ins1, 2, PaymentConfirmationRequestAC)\});$
referring to the variable $xm2$ of transition $Send\_PC\_AC$.

The global formula $(FORALL\_UNTIL(NOT(A_2), A_1)$ holds if the $Payment\ Confirmation$ message is not sent by the Airline Company until the $Payment\ Confirmation\ Request$ has been sent. Note that $InitNode$ means the initial marking of the transition system.

3) Model Checking: Here, we adopt the model checking toolkit provided by CPN Tools 4.0 to check whether the generated behavioral model N meets the two conditions introduced in the behavioral compatibility definition (definition 5).

First, ASK-CTL module should be loaded in CPN Tools 4.0. The commands are shown in high part of Fig. 6. Then,

 ISBN: 978-1-61208-337-7

the SML property description is written and then evaluated by “evaluate ML” option in the simulation tool palette. The checking result is shown in the green part of Fig. 6.

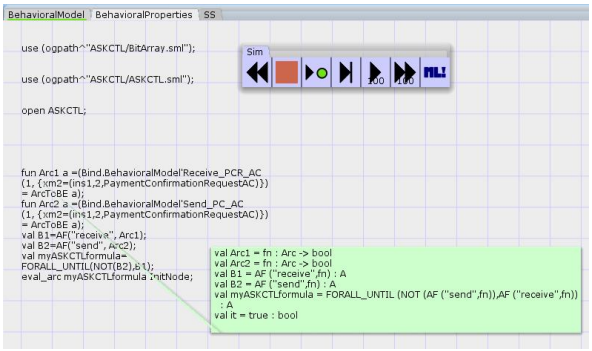


Fig. 6. Model Checking the behavioral property: true

We can see that the checking results returns true, which indicates that the behavioral model satisfies the property. This checking is not sufficient to say that the behavioral model is correct. We also need to check the two conditions given in the behavioral compatibility definition for all pairs (request, answer) in our modeled system. In our approach, if failures are detected then we must return to the first phase to correct these errors. For our example, we make some errors related to message order and in this case the checking results of our same taken property is given in Fig. 7. The correction is based on a behavior failure analysis that is done on exploring all property violation scenarios and pinpoints areas where modeling changes or revisions will be considered.

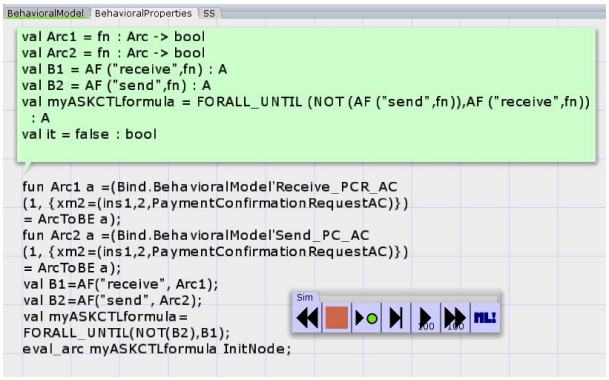


Fig. 7. Model Checking the behavioral property: false

Having shown that CPN based model checking of order property is feasible, we can then exploit the CPN Tools advanced graphical environment, to interactively simulate the actions performed in possible property violation scenarios. Behavior failure analysis is based on inspection of the terminal markings in all property violation paths. The simulation control functionality found in the CPN Tools 4.0 allows firing transitions with an interactively chosen transition. Thus, the actions included in the scenario of interest are easily reproduced and we can explore all possible behavior revision prospects to repair the detected property violation.

V. Related Work

To capture the behavior of service composition in some formal way, a variety of formal analysis techniques have been proposed. Most of them adopt a formal model such as PNs or Finite State Machines (FSM) or pi-calculus to express service behavior in a service orchestration and then utilize its theories and tools to accomplish the automatic verification. For example, Lucchi and Mazzara [16] propose an approach that analyzes service orchestration using WS-BPEL and the formalism pi-calculus. Benatallah et al. [17] propose an approach that analyzes the behavioral compatibility and the similarity of Web services. Hamadi et al. [18] propose an algebra of PNs to analyze the behavioral compatibility of Web services. The orchestration is modeled by the use of simple operators such as arbitrary sequence and more complex operators like iteration. Also, Tan et al. [19] propose an approach to analyze the compatibility of two services by translating their BPEL abstract processes into CPNS and check if their composition violates the constraints imposed by either side.

Compared to the works listed above, the approach proposed in [20] verifies service choreography by checking not only deadlock-freeness but also other properties, such as liveness and other specific properties. This approach is based on the automata formalism for modeling and on model checking for the analysis of behavioral compatibility and the satisfaction of temporal constraints: timing conflicts that may arise in a choreography can be detected. Another example that investigated choreography is [21], where Martnes et al. propose a PN based approach to model and analyze the behavioral compatibility of Web services, initially described by BPEL processes. Each selected BPEL process is transformed into a BPN. Then, the corresponding BPN models are composed, and the deadlock-freeness of the resulting net has to be proven.

In contrast to these works, our paper focuses not only on automatically reasoning about deadlock freeness, but also on message ordering properties. In addition, our verification is done at design time while current approaches are specific to a given programming language and only focus on the verification of already implemented composite services. The benefit of our approach is that the composition specification is proven to be correct before its implementation with a programming language such as BPEL. Few works has been done, to the best of our knowledge, in this research direction. For example, Achilleos et al. [22] propose an approach that combines Model Driven Architecture (MDA) and PNs to provide design, verification and code generation. Recently, in [23], a MDA for creating consistent service orchestrations is presented. Service execution and interaction are described with a high-level model in terms of extended PNs notation. Also, recently, Dumez et al. [24] propose a MDA approach to specify, verify and implement service composition using existing specification and implementation languages. To support the formal verification of the composition, a translation of the composition workflow model is done into a Language

 ISBN: 978-1-61208-337-7

of Temporal Ordering Specification (LOTOS [25]) formal specification. The CADP [26] tool-set is then used to verify the composition via its LOTOS specification. Our work has a similar objective, adopting instead CPNs to formalize the behaviors and interactions of services. This model is well suited to specify service composition due to its compositionality properties. Moreover, it uses CPN Tools, providing the designer with the ASK-CTL toolkit that is expressive enough to describe message ordering.

Our paper presents a formal approach that goes beyond checking for deadlock-freeness as proposed by the majority of related work. We note that our approach has a disadvantage since it is based on state space analysis that presents the *state explosion problem*. To address this problem, we can use reduction techniques that are supported by CPN Tools 4.0.

VI. Conclusion and Future Work

CPNs enhance classical PNs with commonly agreed upon extensions such as data and hierarchy. The resulting modeling language is highly expressive and is supported by CPN Tools 4.0, a recent powerful software tool for the modeling and analysis of CPNs. This paper used an example to explain the behavioral compatibility that is analyzed using CPNs during the early design phase of choreography, thus avoiding iterative cycles between the choreography implementation and the compatibility analysis. The interest of our proposed approach lies in the clear presentation of the analysis model and readiness for its implementation. We have demonstrated how to use CPNs to model and compose the Web service behaviors and how to use CPN Tools 4.0 to analyze their behavioral compatibility basing on ASK-CTL and model checking.

In future work, we will extend our model to allow the performance analysis in terms of quantitative timing constrains. Our new timed model will be based on Timed CPN [7], also supported by CPN Tools 4.0, and its analysis will be done by simulations that allow performance analysis. In addition, we plan to extend our approach of verification by using MDA in order to have a development process of service choreography that is based of CPNs and Timed CPNs.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: A Research Roadmap," *International Journal of Cooperative Information Systems*, vol. 17, no. 02, pp. 223–255, 2008, ISSN: 0-2-1-8-8-4-3-0.

[2] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, 2005, ISBN: 013-14-88-74-0.

[3] Y. Zhu and H. Gao, "A Novel Approach to Generate the Property for Web Service Verification from Threat-Driven Model." *Applied Mathematics & Information Sciences*, vol. 8, no. 2, 2014.

[4] G. Piccinelli, W. Emmerich, C. Zirpins, and K. Schutt, "Web service interfaces for inter-organisational business processes an infrastructure for automated reconciliation," in *Enterprise Distributed Object Computing Conference*. IEEE, 2002, pp. 285–292.

[5] Y. Hongli, Z. Xiangpeng, Q. Zongyan, P. Geguang, and W. Shuling, "A formal model for web service choreography description language (ws-cdl)," in *IEEE International Conference on Web Services, Chicago, IL, USA*. Plattner Institute, University of Potsdam, German, and Queensland University, 2006, pp. 893–4.

[6] M. De Backer, M. Snoeck, G. Monsieur, W. Lemahieu, and G. Dedene, "A Scenario-Based Verification Technique to Assess the Compatibility of Collaborative Business Processes," *Data & knowledge engineering*, vol. 68, no. 6, pp. 531–551, 2009.

[7] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer Berlin Heidelberg, 2009.

[8] A. Cheng, S. Christensen, and K. Mortensen, "Model Checking Coloured Petri Nets-Exploiting Strongly Connected Components," *DAIMI Report Series*, vol. 26, no. 519, 1997.

[9] M. Wastergaad, "CPN tools 4: Multi-Formalism and Extensibility," in *Application and Theory of Petri Nets and Concurrency*. Springer, 2013, pp. 400–409, Jose-Manuel, C. and Jorg, D., Ed., ISBN: 978-36-42-38-69-61, ISSN: 0-3-0-2-9-7-4-3.

[10] Z. Maamar, B. D., G. Mostefaoui, S. Subramanian, and Q. Mahmoud, "Toward Behavioral Web Services Using Policies," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 38, no. 6, pp. 1312–1324, 2008, ISSN: 1-0-8-3-4-4-2-7.

[11] W. M. P. Van der Aalast, C. Stahl, and M. Wastergaard, "Strategies for Modeling Complex Processes Using Colored Petri Nets," in *Transactions on Petri Nets and Other Models of Concurrency VII*. Springer, 2013, pp. 6–55, Jensen, K. and Van der Aalast, W. M. P. and Balbo, G. and Koutny, M. and Wolf, K., Ed., ISBN: 978-36-42-38-14-23, ISSN: 0-3-0-2-9-7-4-3.

[12] S. Christensen and K. Mortensen, "Design/CPN ASK-CTL manual," *University of Aarhus. 0.9 edn*, 1996.

[13] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 8, no. 2, pp. 244–263, 1986.

[14] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual*. Pearson Higher Education, 2004.

[15] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 213–254, 2007, ISSN: 1-4-3-3-2-7-7-9.

[16] R. Lucchi and M. Mazzara, "A pi-calculus based semantics for ws-bpel," *The Journal of logic and Algebraic Programming*, vol. 70, no. 1, pp. 96–118, 2007.

[17] B. Benatallah, C. F., and F. Toumani, "Analysis and Management of Web Service Protocols," in *Conceptual Modeling–ER 2004*, 2004, pp. 524–541, Atzeni, P. and Chu, W. and Lu, H. and Zhou, S. and Ling, T.W., Ed., ISBN: 978-35-40-23-72-35, ISSN: 0-3-0-2-9-7-4-3.

[18] R. Hamadi and B. Benatallah, "A Petri net-based model for web service composition," in *Proceedings of the 14th Australasian database conference-Volume 17*. Australian Computer Society, Inc., 2003, pp. 191–200, ISBN: 090-99-25-95-X.

[19] W. Tan, Y. Fan, and M. Zhou, "A petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language," *Automation Science and Engineering, IEEE Transactions on*, vol. 6, no. 1, pp. 94–106, 2009, ISSN: 1-5-4-5-5-9-5-5.

[20] N. Guermouche and C. Godart, "Characterizing Compatibility of Timed Choreography," *International Journal of Web Services Research*, vol. 8, no. 2, pp. 1–28, 2011.

[21] A. Martnes, S. Moser, A. Gerhardt, and K. Funk., "Analyzing Compatibility of Bpel Processes," in *Proceedings of the International Conference on Internet and Web Applications and Services/Advanced International Conference, 2006*. IEEE, 2006, pp. 147–147, ISBN: 076-95-25-22-9.

[22] A. Achilleos, K. Yang, N. Georgalas, and M. Azmoodech, "Pervasive Service Creation Using a Model Driven Petri Net Based Approach," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*. IEEE, 2008, pp. 309–314.

[23] G. Grossmann, M. Schrefl, and M. Stumptner, "Design for service compatibility," *Software & Systems Modeling*, vol. 12, no. 3, pp. 489–515, 2013, ISSN: 1-6-1-9-1-3-6-6.

[24] C. Dumez, M. Bakhouya, J. Gaber, M. Wack, and P. Lorenz, "Model-Driven Approach Supporting Formal Verification for Web Service Composition Protocols," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1102–1115, 2013.

[25] H. Garavel and J. Sifakis, "Compilation and verification of lotos specifications," in *PSTV*, vol. 10, 1990, pp. 359–376.

[26] J. C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu, "CADP a Protocol Validation and Verification Toolbox," in *Computer Aided Verification*. Springer, 1996, pp. 437–440.

 ISBN: 978-1-61208-337-7

Always stay agile!

Towards Service-oriented Integration of Business Process and Business Rules Management

Christopher Gäth, Alexander Hödicke, Sophia Marth, Jörn Siedentopf, Andreas Hausotter, and Arne Koschel
Hochschule Hannover – University of Applied Science and Arts
Competence Center Information Technology and Management
Hanover, Germany
christopher.gaeth@stud.fh-hannover.de, alexander.hoedicke@stud.fh-hannover.de, sophia.marth@stud.fh-hannover.de,
joern.siedentopf@stud.fh-hannover.de, andreas.hausotter@hs-hannover.de, and arne.koschel@hs-hannover.de

***Abstract*— To keep their competitive edge, enterprises need to change their operational processes in a flexible and agile manner. A Service-oriented Architecture (SOA) may help to meet these needs. One key feature of a SOA is the externalisation of business process logic. However, process logic is often complex, hard to understand and difficult to adapt. This issue is due to a mingling of process and decision logic. In order to ensure flexibility and agility, decision logic should be moved to a separate service. There are several approaches to realise such a "rule" service conceptually. In this paper, a decision framework to select the appropriate rules execution approach is developed, based on a set of "factors". The decision framework is applied to an application scenario from the insurance domain.**

Keywords - Business Process Management (BPM); Business Rules Management (BRM); Business Rules Management System (BRMS); Service-oriented Architecture (SOA); Workflow Management System (WfMS)

I. INTRODUCTION

Workflow Management Systems (WfMS) support companies in the management and execution of business processes [1]. Nowadays, the latest challenges for insurance companies such as the dynamic business environment and compliance with legal requirements highlight the need for business agility [2][3][4]. Business agility requires the individual, quick, and flexible composition and adaption of business processes [5][6]. This can be done in the context of Business Process Management (BPM). As a result of the composition and adaption, the number of decisions may rise within the processes. Hence, the complexity of the business processes can lead to a lack of business agility [4][5].

Business rules provide an opportunity to reduce the complexity of the processes, whilst the complex decision logic is encapsulated. The necessary changes with respect to agility often relate to the complex decision logic and not to the process or business logic. Thus, the separation of decision logic and process logic on the modelling and implementation level is a useful approach to reduce complexity [4].

Comprehensive service-oriented approaches have potential to create business agility [7]. Thus, a Service-oriented Architecture (SOA) can help to address challenges like the dynamic business environment. The service-oriented integration of BPM and Business Rules Management (BRM) provides potential to change business processes in an agile manner [4]. The results of interviews with experts of the insurance service sector emphasised the issue to choose an adequate approach to automate the execution of business rules with respect to a missing decision support. Considering the dynamic business environment in the insurance services sector, the topics of the presented work are of potential value for several insurance companies in Germany [2][3].

The aim and the major contribution of this paper is a decision framework for choosing an adequate business rules execution approach. The result of the application is based on a characteristic application scenario for companies operating in the insurance services industry.

The subsequent research activity is to develop a prototypically service-oriented integration of BPM and BRM based on the results of the applied decision framework.

Thus, the paper illustrates the work in progress related to the current research activities of the Competence Center Information Technology and Management (CC_ITM).

The remainder of the paper is structured as follows. In the first part (section II), the prior and related work are presented. The main sections (III to V) refer to the description of the service-oriented approach based on a workflow engine, followed by an introduction of the application scenario as well as a description and an application of the developed decision framework. Finally, the paper ends with a conclusion and looks upon some future research activities (section VI).

II. PRIOR AND RELATED WORK

Potential application scenarios regarding the combination of BPM and BRM were analysed in accordance with the requirements of the CC_ITM collaboration partners [8]. As a result, the application scenario "handle a goodwill request" was selected. The scenario, introduced in this paper in section IV, is inspired by the insurance application architecture of the General Association of the German Insurance Industry. The insurance application architecture describes inter alia reference process models for the insurance services in Germany [9]. The application scenario was already implemented in prior research projects. The scenario was used for the evaluation of the prototypic implementation regarding a

 ISBN: 978-1-61208-337-7

service-oriented approach based on a workflow engine [10][11][12].

In addition, the elements, which are to be implemented with a rule-based approach, were determined within the scenario. In the process, the business rule "set goodwill adjustment" was identified. [13] suggests an extraction process for business rules identification from business process models. This process is useful, because business rules are often not explicitly included in the process models. A decision guideline for distinguishing between business process and business rule is presented in [14]. Requirements concerning business rules technologies are defined in [2][3]. [15] illustrates variables for determining suitable solutions for business rule implementation. As a result of the literature review, the decision guideline, the requirements and the variables provide a contribution to the decision framework. Since no previous research allows a simple choosing of an adequate business rules execution approach this decision framework is the first to extend the current state of research through the linking of factors, indicators and business rules execution approaches. The determination of the specific business rules execution approach depends on the elements, which are to be implemented with a rule-based approach.

III. SOA APPROACH BASED ON A WORKFLOW ENGINE

Figure 1 illustrates the simplified concept of an already implemented SOA. The architecture comprises several components, which were developed in the context of Business Process Management, Business Activity Monitoring, and Service-oriented Architecture within the prior research projects and implements the application scenario "handle a goodwill request". Based on the results of this research activity, the architecture is extended towards the service-oriented integration of BPM and BRM by a corresponding business rules execution approach.

The workflow engine is used for the coordination and execution of workflow models. Thus, the engine is responsible for ensuring that every activity within the workflow definition is executed in the defined order and that the data flow between the workflow client and the invoked service is routed correctly.

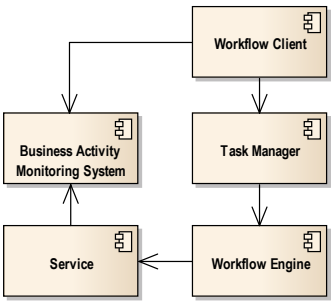


Figure 1. Overview of the architecture

The workflow client presents a user interface to call and activate the workflow. Users take their individual tasks from a "task list". Some activities require input from the user. Therefore, the engine will request the input from the client as a result of a client call.

The task manager is used to decouple the workflow engine and the workflow client. Thus, the task manager is a software component, which abstracts from a specific workflow engine. The decoupling is necessary because the technically tight coupling to a specific workflow engine limits the opportunity of flexibility in the event of an enterprise merger or acquisition [11].

The invoked service implements the business logic necessary to execute the activities specified within the workflow model. In accordance with the defined process model, the workflow engine invokes the service to execute the activity. The invoked service performs the activity and returns the result to the workflow engine.

The Business Activity Monitoring System (BAMS) represents the software component, which is used for the real-time monitoring of critical performance indicators to improve the speed and effectiveness of business functions. The required data are collected within a data warehouse by the performance of extract, transform, and load processes. Based on these data, the BAMS generates complex events to build critical performance indicators. The complex event processing is realised by triggers and stored procedures in active databases. The events are analysed and reported by the BAMS [10][11].

IV. APPLICATION SCENARIO

Figure 2 illustrates the application scenario "handle a goodwill request". The scenario constitutes a part of the simplified insurance process "claim processing" and is required to apply the decision framework to choose an adequate business rules execution approach. In the future, the scenario will be used to evaluate the prototypic implementation.

With the handling of the process "handle a goodwill request", insurance companies check whether and in what amount the customer claims are to be satisfied without obligation of the insurance companies. The task "check goodwill" checks whether a claim without obligation of the insurance company should be regulated in order to not compromise the business relationship. In particular the privity of contract and the relation with the customer are checked. The amount of goodwill will be determined within the business rule "set goodwill adjustment". The last task checks whether a contractual alteration is appropriate for the policyholder. As a result, it may also be determined that no contractual alteration is necessary or that a reasoned recommendation to contractual design shall be given [9].

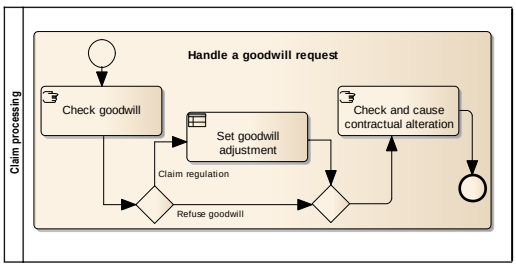


Figure 2. Application scenario "handle a goodwill request"

 ISBN: 978-1-61208-337-7

V. DECISION FRAMEWORK

To define the corresponding execution approach for the identified business rule "set goodwill adjustment" a decision framework was determined. A decision framework was created based on factors gained by the performed literature review. Table I describes the used factors.

TABLE I. FACTORS OF THE DECISION FRAMEWORK

Factor	Definition
Frequency of rule change	The volatility can be seen as a measure of flexibility in terms of changing business rules. To determine the expected volatility of a business rule, changes in the past can be considered.
Understanding of implications	Understanding of implications regarding a business rule modification describes whether the potential impact can be safely predicted or not. The risk level of changes is reflected by this factor.
Distribution time	Distribution time defines the time the business rule needs to give effect. Thus, the time between the release and the business rule provision is defined.
Transparency	The factor "transparency" specifies the need for justification of decisions regarding the execution of a business rule.
Transaction volume	The factor "transaction volume" defines the required volume of facts for rule execution.
Versioning	Most business rules are revised after some time. Here it is important to determine whether the old version of the rule must continue to be present, or whether the business rule is simply overwritten and only the new rule has to be available.

There are several approaches to provide the needed business rules execution approach. [2] identifies an inference machine, a database management system, a business application, and configurations as possible approaches to execute business rules. The decision framework can help in choosing one approach by considering several application scenarios and the factors. Due to economic restrictions in this exemplary implementation only the described application scenario is considered. Every rules execution approach has advantages and disadvantages in regard to the adopted factors.

Inference machines enable the efficient and flexible execution of business rules. The encapsulation of complex decision logic allows an easier and faster implementation of necessary adjustments. An inference machine enables transparency if the engine includes an explanation component. For business rules execution the inference machine requires all relevant facts. This can lead to increased transaction volumes. Most of the inference machines allow rule versioning.

Database management systems are not developed for frequent changes in their structure but triggers, constraints, and stored procedures allow depicting rules. There has to be a high understanding of the implications when changes to databases are made. This leads to a long distribution time. The firing of triggers is often not transparent and changes are often not trackable. Databases are built for data processing and so allow high transaction volumes.

Business applications are built for long-lasting business cases and not for frequent changes. A high understanding of the implications is needed for changing a business application. So, the distribution time for changes made in business applications is long. Business applications need to be built to allow transparency in their decision logic if necessary. Providing several executable versions of business applications is difficult to handle.

Configurations often allow fast modifications but are not designed for frequent changes. As configurations often only allow adjustments in a defined range the understanding of implications is medium. If the configuration is not accessible by the user it may take hours to provide a change. Transparency in the configuration decision logic is only possible through implementation in the business application that accesses the configuration. Versioning is hardly possible for configurations. Table II presents the mapping of indicators to the business rules execution approach.

TABLE II. DECISION FRAMEWORK

Factor	Indicator		
	Business rules execution approach		
Frequency of rule change	High (hourly to weekly)	Low (monthly to annually)	Never
	Inference machine	Configuration / Database	Business Application
Understanding of implications	Low	Medium	High
	Inference machine	Configuration	Database / Business Application
Distribution time	Minutes	Hours	Days
	Inference machine	Configuration	Database / Business Application
Transparency	Yes	n/a	No
	Inference machine	n/a	Database / Configuration / Business Application
Transaction volume	Low	Medium	High
	Inference machine	Business Application / Configuration	Database
Versioning	Yes	n/a	No
	Inference machine	n/a	Configuration / Database / Business Application

The requirements for choosing an adequate business rules execution approach results from the presented application scenario "handle a goodwill request". In this context, a low frequency of rule change is assumed on average. The understanding of implications is high. A distribution time of minutes is expected and there should be transparency. Only a low transaction volume is assumed but there has to be versioning of business rules. The decision framework provides the opportunity to weight the factors. Depending on the respective application scenario the importance of the factors has to be determined. Table III presents the weighting of the factors concerning the application scenario "handle a goodwill request".

 ISBN: 978-1-61208-337-7

TABLE III. IMPORTANCE OF THE FACTORS

Factor	Importance
Frequency of rule change	20%
Understanding of implications	10%
Distribution time	40%
Transparency	10%
Transaction volume	5%
Versioning	15%

The importance of the factors is not generally valid and depends on the individual application scenario. Table IV shows the rating regarding the appropriate business rules execution approach. The rating results from the requirements and the importance of the factors. For example the rating for the inference machine is 70%. Based on the requirements for choosing an adequate business rules execution approach concerning the factors distribution time, transparency, transaction volume, and versioning the inference machine is the result. Regarding the rating result of the inference machine the importance of the factors distribution time (40%), transparency (10%), transaction volume (5%), and versioning (15%) were cumulated.

TABLE IV. RATING RESULTS

Approach	Result
Inference machine	70%
Database	30%
Configuration	20%
Business Application	10%

As explicated above, an inference machine is suggested by the decision framework. The application of the decision framework can be considered as a decision-making aid. In addition to the illustrated factors within the decision framework further factors were identified. To reduce the complexity for choosing an adequate business rules execution approach only factors that are relevant to the application scenario were considered.

VI. CONCLUSION AND FUTURE WORK

The combination of technologies and concepts such as SOA as well as business rules processing / management is a promising approach for companies operating in the insurance services industry. As the key contribution of this article a decision framework for choosing an adequate business rules execution approach was developed and applied. By applying the decision framework one gets a criteria list, which decides when to use a particular concept to implement the business rules approach – especially an inference machine, a DBMS-based approach, a configuration or a hard-coded application. Applying the decision framework to the "handle a goodwill request" scenario results in an inference-machine-based (BRM-based) approach to be most useful. Since the decision framework is so far work-in-progress it will be extended in future work. The future work has to include the evaluation of the usefulness, applicability, and validity of the decision framework and the corresponding results. This could be done by applying the decision framework in practice and determining framework extensions in the context of interviews with experts. Our subsequent research activity is to develop design decisions for the prototypical service-oriented integration of business process management and business rules management regarding the application scenario "handle a goodwill request".

REFERENCES

[1] T. Schäl, "Workflow Management Systems for Process Organisations," Berlin: Springer Verlag, 1996.

[2] M. Schacher and P. Grässle, "Agile Enterprises with Business Rules: The Business Rules Approach," Berlin: Springer Verlag, 2006, ISBN: 978-3540256762.

[3] J. Boyer and H. Mili, "Agile Business Rule Development: Process, Architecture, and JRules Examples," Berlin: Springer Verlag, 2011.

[4] C. Gäth, "Potential of the Business Rules Approach - identifying and exploiting the potential in consideration of supportive and complementary approaches," bachelor thesis, University of Applied Science and Arts, Hanover, unpublished, 2013.

[5] T. van Eijndhoven, M. E. Iacob, and M. L. Ponisio, "Achieving Business Process Flexibility with Business Rules," 12th Int. IEEE Enterprise Distributed Object Computing Conference (EDOC), 2008, pp. 95-104.

[6] M. Döhring, L. Karg, E. Godehardt and B. Zimmermann, "The Convergence of Workflows, Business Rules and Complex Events," 12th Int. Conference on Enterprise Information Systems (ICEIS), 2010, pp. 338-343.

[7] G. Starke and S. Tilkov, "SOA Expert Knowledge," Heidelberg: dpunkt.Verlag, 2007, ISBN: 978-3898644372.

[8] J. Siedentopf, "Concept for a BRM-System and prototypical implementation of an application scenario," bachelor thesis, University of Applied Science and Arts, Hanover, unpublished, 2013.

[9] General Association of the German Insurance Industry, "The application architecture of the insurance industry". [Online]. Available from: http://www.gdv-online.de/vaa/vaafe_html/dokument/psl.pdf (2013, Dec. 22).

[10] T. Bergemann, A. Hausotter, and A. Koschel, "Keeping Workflow-Enabled Enterprises Flexible: WfMS Abstraction and Advanced Task Management," 4th Int. Conference on Grid and Pervasive Computing Conference (GPC), 2009, pp. 19-26.

[11] A. Hausotter, C. Kleiner, A. Koschel, D. Zhang, and H. Gehrken, "Always Stay Flexible! WfMS-independent Business Process Controlling in SOA," 15th IEEE Int. Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011, pp. 184-193.

[12] A. Hödicke, "Conception and Prototypical Implementation of a Software Architecture for a BPM and BRM System," master thesis, University of Applied Science and Arts, Hanover, unpublished, 2013.

[13] O. Levina, O. Holschke, J. Rake-Revelant, "Extracting Business Logic from Business Process Models," 2nd IEEE Int. Conference on Information Management and Engineering (ICIME), 2010, pp. 289-293.

[14] M. Zur Muehlen, M. Indulska, and K. Kittel, "Towards Integrated Modeling of Business Processes and Business Rules," 19th Australasian Conference on Information Systems (ACIS), 2008, pp. 690-697.

[15] M. L. Nelson, R. L. Rariden, and R. Sen, "A Lifecycle Approach towards Business Rules Management," 41st Hawaii Int. Conference on System Sciences (HICCS), 2008, pp. 113-123.

 ISBN: 978-1-61208-337-7

Genetic Algorithm to the Power of SMT: a Hybrid Approach to Web Service Composition Problem

Artur Niewiadomski
Institute of Computer Science
Siedlce University
Siedlce, Poland
e-mail: artur.niewiadomski@uph.edu.pl

Wojciech Penczek
Institute of Computer Science
Polish Academy of Science
Warsaw, Poland
e-mail: penczek@ipipan.waw.pl

Jaroslaw Skaruz
Institute of Computer Science
Siedlce University
Siedlce, Poland
e-mail: jaroslaw.skaruz@uph.edu.pl

***Abstract*—The paper deals with the concrete planning problem – a stage of the Web Service Composition in the PlanICS framework, which consists in choosing the best service offers in order to satisfy the user query and to maximize the quality function. We introduce a novel planning technique based on a combination of a Genetic Algorithm with a Satisfiability Modulo Theories Solver, which allows to obtain better results than each of the methods separately. The paper presents some preliminary, although very encouraging, experimental results.**

***Keywords*-Web Service Composition; Concrete Planning; Genetic Algorithm; Satisfiability Modulo Theories; Hybrid Algorithm**

I. INTRODUCTION

One of the fundamental ideas of Service-Oriented Architecture (SOA) [1] is to compose simple functionalities, accessible via well-defined interfaces, in order to realize more sophisticated objectives. The problem of finding such a composition is hard and known as the Web Service Composition (WSC) problem [1][2][3].

PlanICS [4] is a framework aimed at WSC, easily adapting existing real-world services. The main assumption in PlanICS is that all the web services in the domain of interest as well as the objects that are processed by the services, can be strictly classified in a hierarchy of *classes*, organised in an *ontology*. Another key idea is to divide the planning into several stages. The first phase deals with *classes of services*, where each class represents a set of real-world services, while the other phaces work in the space of *concrete services*. The first stage produces an *abstract plan* composed of service classes [5]. Next, offers are retrieved by the Offer Collector (OC), a module of PlanICS, and used in the concrete planning (CP). As a result of CP, a *concrete plan* is obtained, which is a sequence of offers satisfying predefined optimization criteria. Such an approach enables to reduce dramatically the number of web services to be considered, and inquired for offers.

This paper deals with the Concrete Planning Problem (CPP), shown to be NP-hard [6]. Our previous works employ several techniques to solve it: a Genetic Algorithm (GA) [7], numeric optimization methods [8] as well as Satisfiability Modulo Theories (SMT) Solvers [6]. The results of the extensive experiments show that the proposed methods are complementary, but every single one suffers from some disadvantages. The main disadvantage of an SMT-based solution is often a long computation time, which is not acceptable in the case of a real-world interactive planning tool. On the other hand, a GA-based approach is relatively fast, but it yields solutions, which are far from optimum and found with a low probability.

Thus, our aim is to exploit the advantages of both methods by combining them into one hybrid algorithm, which is the main contribution of this paper. The main idea of our new hybrid approach involves a modification of the standard GA, such that after every couple of iterations of GA, several top-ranked individuals are processed by the SMT-based algorithm in order to improve them.

Over the last few years, CPP has been extensively studied in the literature. G. Canfora et al. [9] use a simple GA to obtain a good quality concrete plan. Y. Wu et al. [10] transforms CPP to a multi-criteria optimization problem and exploits GA to find a concrete plan. However, the authors present the experiments on a relatively small search space that could not provide valuable conclusions.

The rest of the paper is structured as follows. In Section II the PlanICS framework is introduced and CPP is defined. Section III presents the main ideas of our hybrid approach as well as some technical solutions. Next, the preliminary experimental results are presented and discussed. The paper ends with some conclusions.

II. CONCRETE PLANNING PROBLEM

This section introduces the main ideas behind the PlanICS framework and gives all the necessary definitions for defining the concrete planning problem.

An ontology contains a system of *classes* describing the types of the services as well as the types of the objects they process. A class consists of a unique name and a set of the attributes. By an *object* we mean an instance of a class. By a *state* of an object we mean a valuation of its attributes. A set of objects in a certain state is called a *world*. A key notion of PlanICS is that of a *service*. We assume that each service processes a set of objects, possibly changing values of their attributes, and produces a set of new (additional) objects. We say that a service *transforms* a world. The types of services available for planning are defined as elements of the branch of

 ISBN: 978-1-61208-337-7

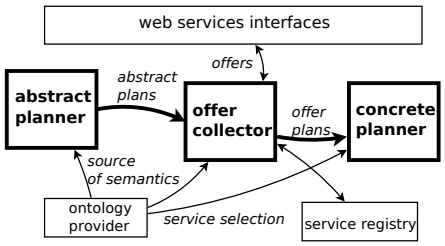


Figure 1. A simplified diagram of the PlanICS system architecture.

classes rooted at *Service* concept. Each service type stands for a description of a set of real-world services of similar functionality.

The main goal of the system is to find a composition of services that satisfies a user query. The query interpretation results in two sets of worlds: the initial and the expected ones. Moreover, the query may include additional constraints, especially *quality constraints*, the sum of which is used to choose the best from all the potential solutions. Thus, the task of the system is to find such a set of services, which transform some initial world into a world matching some expected one in such a way that the value of the quality function is maximized. Figure 1 shows the general PlanICS architecture. The bold arrows correspond to computation of a plan, the thin arrows model the planner infrastructure.

In the first stage of the composition, an *abstract planner* matches services at the level of input/output types and the abstract values. At this planning stage, it is enough to know if an attribute does have a value, or it does not, so we abstract from the concrete values of the object attributes. The result of this stage is a Context Abstract Plan (CAP), consisting of a multiset of service types (defined by a representative sequence), contexts (mappings between services and the objects being processed), and a set of final worlds containing objects that fulfill the user query. See [5] for more details.

In the second planning stage, a CAP is used by OC, i.e., a tool, which in cooperation with the service registry, queries real-world services. The service registry keeps an evidence of real-world web services, registered accordingly to the service type system. During the registration, the service provider defines a mapping between the input/output data of the real-world service and the object attributes processed by the declared service type. OC communicates with the real-world services of types present in a CAP, sending the constraints on the data, which can potentially be sent to the service in an inquiry, and on the data expected to be received in an offer in order to keep on building a potential plan. Usually, each service type represents a set of real-world services. Moreover, querying a single service can result in a number of offers. Thus, we define offer sets as the main result of the second planning stage.

Definition 1 (Offer, Offer set): Assume that the n-th instance of a service type from a CAP processes some number of objects having in total m attributes. A single offer collected by OC is a vector $P = [v_1, v_2, \ldots, v_m]$, where v_j is a value of a single object attribute processed by n-th service of the CAP. An offer set O^n is a $k \times m$ matrix, where each row corresponds to a single offer and k is the number of offers in the set. Thus, the element $o^n_{i,j}$ is the j-th value of the i-th offer collected from the n-th service type instance from the CAP.

The responsibility of OC is to collect a number of offers, where every offer represents one possible execution of a single service. However, other important tasks of OC are: (1) building a set of constraints resulting from the user query and from semantic descriptions of service types, and (2) a conversion of the quality constraints expressed using objects from the user query to an *objective function* built over variables from offer sets. Thus, we can formulate CPP as a constrained optimization problem.

Definition 2 (CPP): Let n be the length of CAP and let $\mathbb{O} = (O^1, \ldots, O^n)$ be the vector of offer sets collected by OC such that for every $i = 1, \ldots, n$

$O^i = \begin{bmatrix} o^i_{1,1} & \ldots & o^i_{1,m_i} \\ \vdots & \ddots & \vdots \\ o^i_{k_i,1} & \ldots & o^i_{k_i,m_i} \end{bmatrix}$, and the j-th row of O^i is denoted by P^i_j. Let $\mathbb{P}$ denote the set of all possible sequences $(P^1_{j_1}, \ldots, P^n_{j_n})$, such that $j_i \in \{1, \ldots, k_i\}$ and $i \in \{1, \ldots, n\}$. The Concrete Planning Problem is defined as:

$$max\{Q(S) \mid S \in \mathbb{P}\} \text{ subject to } \mathbb{C}(S), \qquad (1)$$

where $Q : \mathbb{P} \mapsto \mathbb{R}$ is an objective function defined as the sum of all quality constraints and $\mathbb{C}(S) = \{C_j(S) \mid j = 1, \ldots, c$ for $c \in \mathbb{N}\}$, where $S \in \mathbb{P}$, is a set of constraints to be satisfied.

Finding a solution of CPP consists in selecting one offer from each offer set such that all constraints are satisfied and the value of the objective function is maximized. This is the goal of the third planning stage and the task of a concrete planner.

Example. Consider a simple ontology describing a fragment of some financial market consisting of service types inheriting from the class $Investment$, which represent various types of financial instruments. Moreover, the ontology contains three object types: $Money$ having the attribute $amount$, $Transaction$ having the two attributes $amount$ and $profit$, and $Charge$ having the attribute fee. Suppose that each investment service takes m - an instance of $Money$ as input, produces t and c - instances of $Transaction$ and $Charge$, and updates the amount of money remaining after the operation, i.e., the attribute $m.amount$. Assume that the user would like to invest up to \$100 in three financial instruments, but he wants to locate more than \$50 in two investments. Moreover, the user wants to maximize the sum of profits and wants to use only services of handling fees less than \$3. The latter two conditions can be expressed as an appropriate quality function and an aggregate condition. Consider an exemplary CAP consisting of three instances of the $Investment$ service type. A single offer collected by OC is a vector $[v_1, v_2, v_3, v_4, v_5]$, where v_1 corresponds to $m.amount$, v_2 to $t.amount$, v_3 to $t.profit$, and v_4 to $c.fee$. Since the attribute $m.amount$ is updated during the transformation, the offers should contain values from the world before and after the transformation. Thus v_5 stands for the value of $m.amount$ after modification. Assuming that instances of $Investment$ return k_1, k_2, and k_3 offers in response to the subsequent inquiries, we obtain three offer sets: O^1, O^2, and O^3, where O^i is a $k_i \times 5$

matrix of offer values. The conditions from the query are translated to the following constraints: $C_1 := (o^1_{i_1,1} \leq 100)$ and $C_2 := (o^1_{i_1,2} + o^2_{i_2,2} > 50)$, where i_1, i_2, and i_3 are variables ranging over $1 \dots k_i$. Moreover, the amount of money left after the operation is an input for the next investment. Thus, we have: $C_3 := (o^1_{i_1,5} = o^2_{i_2,1})$ and $C_4 := (o^2_{i_2,5} = o^3_{i_3,1})$. The aggregate condition is translated to the following constraint: $C_5 := \big(max(\{o^1_{i_1,4}, o^2_{i_2,4}, o^3_{i_3,4}\}) < 3\big)$, while the quality expression is translated to the quality constraint $Q_1 := \sum_{j=1}^{3} o^j_{i_j,3}$.

III. A Hybrid Solution and Preliminary Results

The analysis of several hard CPP instances is our main motivation to combine the power of SMT with the potential of GA. The main disadvantage of a "pure" SMT-based solution is often a long computation time, which is not acceptable in the case of a real-world interactive planning tool. On the other hand, a GA-based approach is relatively fast, but it yields solutions, which are far from optimum and found with low probability. Thus, our aim is to exploit the advantages of both the methods by combining them into one hybrid algorithm.

A. Overview

The main idea is as follows. The base of our hybrid approach is the standard GA aimed at solving CPP. GA is a non deterministic algorithm maintaining a population of potential solutions during an evolutionary process. A potential solution is encoded in a form of a GA individual, which, in case of CPP, is a sequence of natural values. In each iteration of GA, a set of individuals is selected for applications of genetic operations, such as the standard one-point crossover and mutation, which leads to obtaining a new population passed to the next iteration of GA. The selection of an individual, and thus the promotion of its offspring to the next generation depends on the value of the *fitness function*. The fitness value of an individual is the sum of the optimization objective and the ratio of the number of the satisfied constraints to the number of all the constraints (see Def. 2), multiplied by some constant β:

$$fitness(I) = q(S_I) + \beta \cdot \frac{|sat(\mathbb{C}(S_I))|}{c}, \qquad (2)$$

where I stands for an individual, S_I is a sequence of the offer values corresponding to I, $sat\big(\mathbb{C}(S_I)\big)$ is a set of the constraints satisfied by a candidate solution represented by I, and c is the number of all constraints. The role of β is to reduce both the components of the sum to the same order of magnitude and to control the impact of the components on the final result. The value of β depends on the estimation of the minimal and the maximal quality function value.

The main idea of our new hybrid approach involves the following modification of the standard GA. After every couple of iterations of GA, several top-ranked individuals are processed by the SMT-based algorithm. Given an individual I, the procedure searches for a similar, but improved individual I', which represents a solution satisfying all the constraints and having a greater value of the objective function at the same time. The similarity between I and I' consists in sharing a number of genes. We refer to the problem of finding such an individual as to the *Search for an Improved Individual* (*SFII*).

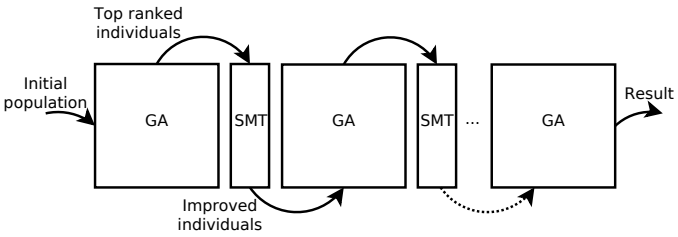


Figure 2. Hybrid algorithm overview.

Since there are many possible ways to exploit this idea, we start from the one which randomly selects the genes to be changed. The overview of our hybrid algorithm is depicted in Figure 2.

The SMT procedure combined with GA is based on the encoding exploited in our "pure" SMT-based concrete planner [6][8]. The idea is to encode *SFII* as an SMT formula which is satisfiable if such an individual exists. First, we initialize an SMT-solver allocating a set $\mathcal{V}$ of all necessary variables:

- $\mathbf{oid}^i$, where $i = 1 \dots n$ and n is the length of the abstract plan. These variables are needed to store the identifiers of offers constituting a solution. A single $\mathbf{oid}^i$ variable takes a value between 1 and k_i.
- $\mathbf{o}^i_j$, where $i = 1 \dots n$, $j = 1 \dots m_i$, and m_i is the number of offer values in the i-th offer set. We use them to encode the values of S, i.e., the values from the offers chosen as a solution. From each offer set O^i we extract the subset R^i of offer values which are present in the constraint set and in the quality function, and we allocate only the variables relevant for the plan.

Next, using the variables from $\mathcal{V}$, we encode the offer values, the objective function, and the constraints, as the formulas shared by all calls of our SMT-procedure. The offer values from the offer sets $\mathbb{O} = (O^1, \dots, O^n)$ are encoded as the formula

$$ofr(\mathbb{O}, \mathcal{V}) = \bigwedge_{i=1}^{n} \bigvee_{d=1}^{k_i} \Big(\mathbf{oid}^i = d \ \wedge \bigwedge_{o^i_{d,j} \in R^i} \mathbf{o}^i_j = o^i_{d,j} \Big). \qquad (3)$$

The formulae $ctr\big(\mathbb{C}(S), \mathcal{V}\big)$ and $qual\big(Q(S), \mathcal{V}\big)$, denoted as $\mathbf{ctr}$ and $\mathbf{q}$ for short, encode the constraints and the objective function, respectively. Details are provided in [6].

Let $I = (g_1, \dots, g_n)$ be an individual, $M = \{i_1, \dots, i_k\}$ the set of indices of genes allowed to be changed, and $q(S_I)$ the value of the objective function where $n, k \in \mathbb{N}$.

Hence, the *SFII* problem is reduced to the problem of satisfiability of the following formula:

$$\bigwedge_{i \in \{1,\dots,n\} \setminus M} (\mathbf{oid}^i = g_i) \wedge ofr\big(\mathbb{O}, \mathcal{V}\big) \wedge \mathbf{ctr} \wedge (\mathbf{q} > q(S_I)) \qquad (4)$$

That is, the formula (4) is satisfiable only if there exists an individual $I' = (g'_1, \dots, g'_n)$ satisfying all the constraints, where $\forall_{i \notin M}\ g_i = g'_i$ and $q(S_{I'}) > q(S_I)$, i.e., sharing with I all genes of indices outside M and having the larger value of objective function than I. If the formula is satisfiable, then

 ISBN: 978-1-61208-337-7

the values of the changed genes are decoded from the model returned by the SMT-solver, and the improved individual I' replaces I in the current population.

B. Experimental Results

As benchmarks for our experiments we choose four instances of CPP, which turned out to be difficult to solve using our "pure" SMT- and GA-based planner [6][8]. All the instances represent plans of length 15. Each offer set of Instances 1 and 3 contains 256 offers, hence, the number of the potential solutions equals $256^{15} = 2^{120}$. In the case of Instances 2 and 4 each offer set consists of 512 offers, thus the size of the search space is $512^{15} = 2^{135}$. The objective functions are as follows:

$$Q_{1,2} = \sum_{i=1}^{n} o^i_{j_i,1}, \qquad Q_{3,4} = \sum_{i=1}^{n} (o^i_{j_i,1} + o^i_{j_i,2}), \qquad (5)$$

while the set of the constraints is the same for all instances, and is defined as:

$$\mathbb{C} = \{(o^i_{j_i,2} < o^{i+1}_{j_{i+1},2})\}, \quad \text{for } i = 1, \ldots, n-1. \qquad (6)$$

Besides the ordinary parameters of GA (which have been set to the same values as in pure GA), that is, the population size (1000), the number of iterations (100), the crossover and mutation probabilities (95% and 0.5% respectively), we introduce also parameters influencing the behaviour of the SMT component. Namely, when to start the SMT procedure for the first time (in the $20th$ iteration), how often the SMT procedure should be run (the parameter **int** stands for the number of the iterations between the subsequent SMT calls), the number of individuals passed to the SMT-solver during one iteration (parameter **inds**), and how many genes are allowed to be changed by SMT (the parameter **ch.genes**). Every instance has been tested 12 times, using a different combination of the parameters combination, and every experiment has been repeated 30 times on a standard PC with 2.8GHz CPU and Z3 [11] version 4.3 as SMT-solving engine.

The preliminary results of applying our new hybrid algorithm to Instance 1 and 2 are presented in Table I, where the columns from left to right display the parameter values and for each Instance, the total runtime of the algorithm (**t[s]**), the average quality of solutions found (**avgQ**), and the probability of finding a solution (**P**). For reference, we report in the two bottom rows the results of the pure SMT- and GA-based planners. One can easily see that quality values obtained in every experiment are higher than these returned by GA. However, in several cases the runtime or probability is not acceptable. We marked in bold the results, where the probability of finding a solution is at least 40% and the runtime is lower than that of the pure SMT-based planner.

For Instances 3 and 4, which objective function is more difficult, the results are given in Table II. Still, in some cases, the results are better than these returned by the pure planning methods. Note that the pure SMT-based algorithm was not able to find the optimal solution within given time limit (500 sec.).

Although the results are encouraging, the hybrid solution is clearly a trade-off between the three measures: the quality, the probability, and the computation time of the pure algorithms. In order to compare the results obtained taking all the measures into account at the same time, we define four simple score functions: $score_i(P, t, avgQ) = \frac{P}{t} \cdot (avgQ - const_i)$, where P, t, and $avgQ$ stand for the probability, the computation time, and the average quality, respectively, and $const_i$ is a parameter, which value is selected in such a way that for each Instance i from I to IV, the score of the pure GA- and SMT-based algorithm is the same. These scores are the benchmarks for the comparison given in Figure 3. The values on the X-axes correspond to the rows of Table I and II, while the Y-axes indicate the values of the score functions. The black bars stand for the best hybrid results in comparison with the pure SMT- and GA-based algorithms. Notice that the hybrid algorithm can improve the solution score of each pure algorithm from 2 times (Instance 4) to nearly 6 times (Instance 1).

TABLE I. EXPERIMENTAL RESULTS FOR INSTANCES 1 AND 2.

Parameters			Instance 1			Instance 2		
ch.genes	**inds**	**int**	**t[s]**	**avgQ**	**P**	**t[s]**	**avgQ**	**P**
8	1	10	9.29	1305.0	3.33	14.94	1382.0	6.67
		20	8.25	1331.5	6.67	13.23	1371.5	13.3
	10	10	**41.04**	**1386.7**	**53.3**	59.52	1437.6	36.7
		20	22.44	1389.0	26.7	41.73	1414.0	33.3
	20	10	**76.29**	**1405.8**	**70.0**	**118.1**	**1441.0**	**73.3**
		20	**34.28**	**1356.5**	**43.3**	**61.94**	**1420.3**	**40.0**
12	1	10	**39.61**	**1363.1**	**66.7**	**56.59**	**1405.3**	**93.3**
		20	**14.48**	**1326.9**	**46.7**	**20.38**	**1380.0**	**40.0**
	10	10	**203.6**	**1417.6**	**100**	**273.2**	**1455.8**	**100**
		20	**114.7**	**1362.2**	**100**	**155.9**	**1431.3**	**100**
	20	10	346.5	1424.2	100	443.1	1460.5	100
		20	**196.4**	**1416.5**	**100**	**261.7**	**1455.3**	**100**
Pure SMT			266	1443	100	388	1467	100
Pure GA			4.96	1218.5	8	5.61	1319.9	10

TABLE II. EXPERIMENTAL RESULTS FOR INSTANCES 3 AND 4.

Parameters			Instance 3			Instance 4		
ch.genes	**inds**	**int**	**t[s]**	**avgQ**	**P**	**t[s]**	**avgQ**	**P**
8	1	10	13.05	2176.5	6.67	22.08	2229.5	6.67
		20	12.36	2054.3	10.0	22.02	2193.6	16.7
	10	10	**121.7**	**2311.5**	**46.7**	**248.3**	**2359.1**	**43.3**
		20	54.18	2279.4	26.7	**151.9**	**2353.5**	**43.3**
	20	10	**324.9**	**2369.4**	**76.7**	566.8	2390.7	60.0
		20	**175.7**	**2304.2**	**50.0**	**290.8**	**2334.1**	**53.3**
12	1	10	**208.1**	**2153.4**	**46.7**	**239.7**	**2216.3**	**56.7**
		20	54.05	2274.1	36.7	64.08	2167.0	26.7
	10	10	1727	2377.9	100	2205	2485.3	100
		20	1066	2327.7	96.7	1325	2414.3	96.7
	20	10	2814	2447.1	100	4456	2568.2	100
		20	2027	2387.3	100	2477	2469.8	96.7
Pure SMT			500	2266*	100	500	2409*	100
Pure GA			5.95	2085.4	10	6.64	2001.9	7

IV. CONCLUSION AND FUTURE WORK

The prototype of the hybrid concrete planner has been implemented and some preliminary experiments have been performed. The very first results show that even using a simple, or a naive strategy of combining the SMT- and GA-based approach, one can obtain surprisingly good results. We believe that the proposed method has a big potential. We plan to further improve the efficiency of our hybrid approach in terms of: a better quality of solutions, lower computation times, as well as higher probabilities of finding solutions. Another important task to be addressed in a future work is to investigate how to choose the parameter values, in order to get a trade-off between quality, probability, and the computation time desired by the user. Moreover, using the experience gained from the concrete

 ISBN: 978-1-61208-337-7

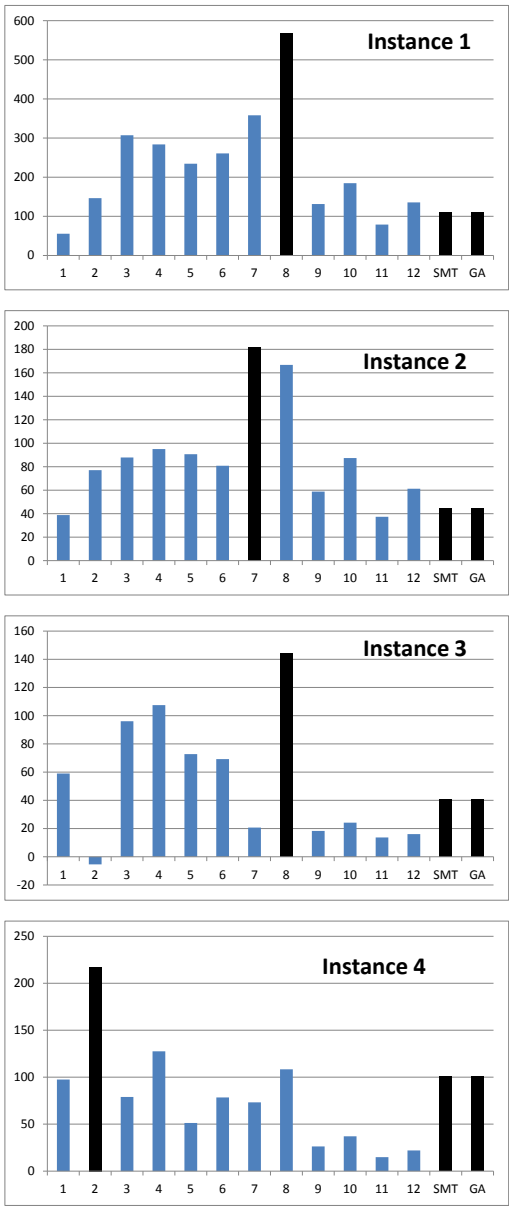


Figure 3. The evaluation of the experimental results using the score functions.

planning, we intend also to develop a hybrid solution for the abstract planning stage.

ACKNOWLEDGMENT

This work has been supported by the National Science Centre under the grant No. 2011/01/B/ST6/01477.

REFERENCES

[1] M. Bell, Introduction to Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. John Wiley & Sons, 2008.

[2] S. Ambroszkiewicz, "Entish: A language for describing data processing in open distributed systems," Fundam. Inform., vol. 60, no. 1-4, 2003, pp. 41–66.

[3] J. Rao and X. Su, "A survey of automated web service composition methods," in Proc. of SWSWPC'04, ser. LNCS, vol. 3387. Springer, 2005, pp. 43–54.

[4] D. Doliwa et al., "PlanICS - a web service compositon toolset," Fundam. Inform., vol. 112(1), 2011, pp. 47–71.

[5] A. Niewiadomski and W. Penczek, "Towards SMT-based Abstract Planning in PlanICS Ontology," in Proc. of KEOD 2013 International Conference on Knowledge Engineering and Ontology Development, September 2013, pp. 123–131.

[6] A. Niewiadomski, W. Penczek, and J. Skaruz, "Smt vs genetic algorithms: Concrete planning in planics framework," in CS&P, 2013, pp. 309–321.

[7] J. Skaruz, A. Niewiadomski, and W. Penczek, "Automated abstract planning with use of genetic algorithms," in GECCO (Companion), 2013, pp. 129–130.

[8] A. Niewiadomski, W. Penczek, J. Skaruz, M. Szreter, and M. Jarocki, "SMT versus Genetic and OpenOpt Algorithms: Concrete Planning in the PlanICS Framework," (submitted to Fundam. Inform.), 2014.

[9] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, 2005, pp. 1069–1075.

[10] Y. Wu and X. Wang, "Applying multi-objective genetic algorithms to qos-aware web service global selection," Advances in Information Sciences and Service Sciences, vol. 3(11), 2011, pp. 134–144.

[11] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in Proc. of TACAS'08, ser. LNCS, vol. 4963. Springer-Verlag, 2008, pp. 337–340.

 ISBN: 978-1-61208-337-7

Towards a Flexible and Privacy-Preserving Reputation System for Markets of Composed Services

Sonja Brangewitz
Department of Economics
University of Paderborn
Paderborn, Germany
Email: sonja.brangewitz@wiwi.upb.de

Ronald Petrlic
Network Security Group
University of Paderborn
Paderborn, Germany
Email: ronald.petrlic@upb.de

Alexander Jungmann
C-LAB
University of Paderborn
Paderborn, Germany
Email: alexander.jungmann@c-lab.de

Marie C. Platenius
Heinz Nixdorf Institute
University of Paderborn
Paderborn, Germany
Email: m.platenius@upb.de

***Abstract*—One future goal of service-oriented computing is to realize global markets of composed services. On such markets, service providers offer services that can be flexibly combined with each other. However, most often, market participants are not able to individually estimate the quality of traded services in advance. As a consequence, even potentially profitable transactions between customers and providers might not take place. In the worst case, this can induce a market failure. To overcome this problem, we propose the incorporation of reputation information as an indicator for expected service quality. We address *On-The-Fly Computing* as a representative environment of markets of composed services. In this environment, customers provide feedback on transactions. We present a conceptual design of a reputation system which collects and processes user feedback, and provides it to participants in the market. Our contribution includes the identification of requirements for such a reputation system from a technical and an economic perspective. Based on these requirements, we propose a flexible solution that facilitates the incorporation of reputation information into markets of composed services while simultaneously preserving privacy of customers who provide feedback. The requirements we formulate in this paper have just been partially met in literature. An integrated approach, however, has not been addressed yet.**

Keywords–Reputation; Service Market; Service Composition; Privacy Protection; On-The-Fly Computing.

I. INTRODUCTION

A major goal of On-The-Fly (OTF) Computing [1][2][3] is the automated composition of software services that are traded on dynamic markets and that can be flexibly combined with each other. A user formulates a request for an individual software solution, receives an answer in terms of a composed service, and finally executes the composed service.

As an illustrative example, let us assume that someone wants to post-process a holiday video. However, it does not pay off to use a monolithic software solution because such software provides a lot of dispensable functionality, and is therefore too expensive to buy for just this purpose. What this person needs is an individually customized software composed of only those services, which together are able to satisfy his needs. A famous web-based platform for individual post-processing tasks is Instagram [4], which provides different image processing services that can be applied to an uploaded photo or video. However, the variety of available services is restricted and the selection of appropriate services has still to be done manually.

Now, let us consider a market of image processing services. A person, who wants to post-process his video, becomes a user within this market by formulating a request describing what he expects from the composed service (e.g., the functionality to create videos with reduced image noise and an increased brilliance homogeneously distributed throughout the entire video). Subsequently, a post-processing solution that satisfies the user's request is automatically composed based on image processing services that are supplied by different market participants. In this scenario, the user only has to pay for the actually utilized functionality.

However, for market participants it is difficult to estimate the quality of services before the service is actually used. For example, an image processing service's response time can be predicted to a certain extent, but it is very dependent on the specific context, e.g., its execution environment and its current load. Other markets such as eBay or Amazon solve this problem by using a reputation system. Within such a system, the experiences other users made in previous transactions are collected. Thereby, the reputation information provides new users an indicator for the service quality they can expect. As an example, let us consider that many users were entirely satisfied with a specific image processing service and rated it with five stars, for example. As a consequence, this service gained a high reputation, which makes it more attractive for future users. Not only the requesters, but also the whole market benefits from considering reputation, because the providers of high-quality products are rewarded with a high reputation, thereby increasing their chances for future sales. On the other hand, low-quality or even deceptive service providers will vanish from the market after some time, which again pays off for all customers. Existing reputation systems used by eBay or

 ISBN: 978-1-61208-337-7

Amazon, for example, do not explicitly consider ratings for composed services. Other reputation systems, such as those to rate trips or hotels, often ask the user to evaluate different aspects. However, single services cannot be combined with each other as flexibly as needed on the OTF market. Thus, a reputation system for composed services is still an open challenge.

The contribution of this paper covers the identification of requirements for a reputation system for markets of composed services such as OTF Computing. Furthermore, it covers the conceptual design of our proposed solution in terms of a flexible reputation system. Technical details and intermediate results of a prototypical implementation are not part of the contribution and are consequently beyond the scope of this paper. We are, however, currently working on an exemplary realization in order to analyze the influence and demonstrate the benefit of the incorporation of reputation information into the OTF Computing process. The contribution of this paper is not necessarily restricted to OTF Computing alone. Results of our work can also be adopted to other areas in which reputation of combinable products play a role.

To the best of our knowledge, there are currently no existing reputation system approaches that can be directly applied in OTF Computing. There are indeed reputation systems which cover the requirement of privacy protection. However, either those systems entail a high overhead and are thus impractical (as covered in related work) and too inflexible to be used in such a complex scenario as in OTF Computing, or privacy is only a "property" which is said to be achieved—but not enforced cryptographically. We rather pursue a *privacy-by-design* approach for our proposed reputation system in OTF Computing. Related to our idea of flexibility, reputation provided and requested depending on specific circumstances has been studied in multi-agent systems [5][6]. Furthermore, reputation has already been considered in the area of service composition: A survey is presented by Mármol et al. [7]. However, privacy protection is not considered by already existing approaches. Each of the existing approaches only deals with a subset of the requirements we identified.

This paper is organized as follows. Section II introduces OTF Computing while mainly focusing on those aspects that are relevant for the work at hand. Furthermore, it motivates the significance of reputation in OTF Computing. Section III gives a detailed problem description by subsequently introducing crucial requirements for a reputation system in OTF Computing. Section IV presents our conceptional solution in terms of a flexible reputation system that covers all identified requirements. Existing approaches that only partially cover these requirements are discussed in Section V. Section VI points out remaining research challenges. Finally, the paper concludes with Section VII.

II. On-The-Fly Computing

A major goal of OTF Computing is automated composition of flexibly combinable services that are traded on markets. A user's request for an individual software solution should be resolved by automatically composing a solution on demand. OTF Computing addresses the entire process, starting with fundamental concepts for organizing large-scale service markets up to the final execution of a composed service. Embedding automatic service composition into service markets is one key challenge for realizing OTF Computing.

A. Automatic Service Composition

In general, we interpret automatic service composition as the sequential application of composition steps. A composition step may, for example, correspond to selecting a service in order to realize a placeholder within a workflow [8]. Regarding our initial example in terms of image processing services, a placeholder could correspond to a class of services which provide similar functionality (such as smoothing filters). For execution, a specific service (e.g., Gaussian smoothing) must then to be selected. A composition step, however, may also correspond to a single step within a composition algorithm based on Artificial Intelligence (AI) planning approaches [9][10].

For simplicity, let us assume that a workflow is available and that a service composition step corresponds to selecting a service. We divide a single composition step into two separate processes which subsequently reduce the amount of qualified service candidates. First of all, a *Service Matching* process determines to what extent a particular service fulfills a placeholder's functional (e.g., signatures and behavior) as well as non-functional requirements (e.g., quality properties such as response time or reliability) [11][12]. Based on the matching result, services that provide significantly different functionality or that violate important non-functional restrictions can be discarded directly. Subsequent to the matching process, a *Service Recommendation* process identifies (and ranks) the best service candidate(s) out of the set of remaining services. During the recommendation process, explicitly given non-functional objectives regarding the final composed service (e.g., maximizing the performance while simultaneously minimizing the costs) as well as implicit knowledge from previous composition processes (e.g., a certain service is more qualified in a particular context than others) are incorporated. The incorporation of knowledge from previous composition processes is realized by means of Reinforcement Learning [13] and requires feedback about the quality of the execution result [14].

B. Market Infrastructure Perspective

Figure 1 shows the transactional view on the entire OTF Computing process, reduced to those processes that are relevant for the work at hand. *OTF Provider Selection* and *Service Provider Selection* are decision-making processes regarding transactions within the market. Three different classes of market participants are involved in the overall process: users,

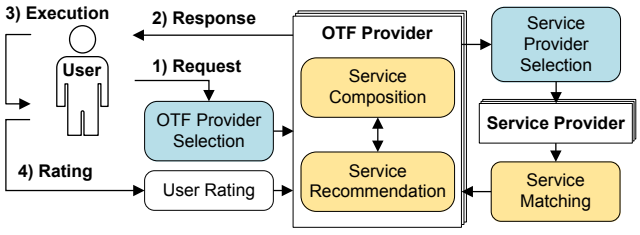


Figure 1: Overall On-The-Fly Computing process.

OTF providers, and service providers. A user formulates a request for an individual software solution and sends it to an OTF provider of his choice (*Step 1*). The selected OTF provider processes the request and automatically composes a solution based on elementary services that are supplied by independent service providers.

For each composition step, an OTF provider asks a selected subset of service providers for elementary services. The previously mentioned matching process is part of the OTF architecture and takes place before an OTF provider receives answers about appropriate elementary services. The matching process operates as a filter ensuring that only services that fulfill the desired requirements to a certain extent are returned. The recommendation process, in turn, is part of the OTF provider-specific composition process and highly depends on the context of the request.

As soon as a composed service is created, it is passed on to the user (*Step 2*), who subsequently executes it (*Step 3*). After execution, the user rates his degree of satisfaction regarding the quality of the execution result (*Step 4*). In the current setting, the value of the user rating is immediately returned to the associated OTF provider. By transforming the value into a reward and incorporating it into the Reinforcement Learning process within the recommendation system, the OTF provider improves his internal composition strategy (recommendation process) for future user requests [15].

C. Reputation as Signal for Quality

In a dynamic market of software services, information about quality (e.g., service quality or the quality of OTF providers) is essential. A user may resort only to OTF providers of a certain quality (e.g., with respect to customer support), while simultaneously accepting only composed services of a certain quality level (e.g., composed services with high reliability and trustworthiness). OTF providers, in turn, have to build composed services consisting of elementary services with a quality level according to a user's request. Information about quality, however, is either difficult to estimate before a transaction actually took place, or cannot be simply trusted if the quality information is provided by the associated market participant itself (e.g., when a service provider specifies the quality of his own services). Our solution to overcome these issues is to replace the previously mentioned and fairly simple user rating procedure (cf. Figure 1) with a flexible reputation system, which aggregates user ratings into single reputation values and provides them to market participants. Reputation can then be incorporated as an estimation of quality into the different decision-making processes.

III. Problem Description and Requirements

Our goal is to explicitly incorporate reputation information as an estimation of quality into the OTF Computing process. Using goal-oriented requirements engineering [16], we systematize our reputation information requirements by investigating the role of reputation from different perspectives.

A. Reputation Information Within the On-The-Fly Process

As shown in Figure 1, the OTF Computing process is initiated by a user's request. To enable users to choose an OTF provider they want to establish a business relationship with, i.e., to buy a composed service from, reputation information about OTF providers must be available.

(*R1*) *OTF Provider Reputation:* The reputation system must provide reputation information about OTF providers.

The selected OTF provider has to ensure that the requested composed service satisfies the user's requirements regarding reputation. For this purpose, the reputation of service providers and the reputation of their supplied elementary services has to be considered during the composition process. In order to enable OTF providers to select service providers they want to retrieve elementary services from, reputation information about service providers must be available.

(*R2*) *Service Provider Reputation:* The reputation system must provide reputation information about service providers.

Reputation of elementary services influences the reputation of composed services. For example, if a composed image processing service uses a well-known, reputable implementation of a specific image filter, it can be assumed, that the composed service's reputation will be higher, than the reputation of a composed service made of unknown elementary services. Thus, the service matching processes (cf. Figure 1) as well as the service recommendation process have to consider the reputation of elementary services. While the matching process has to determine to what extent an elementary service fulfills certain requirements considering reputation, the recommendation process has to determine the best composition steps including reputation. Reputation information, however, cannot be simply extrapolated from service providers to elementary services, since a service provider may supply services of varying quality. Therefore, reputation information about elementary services must be available, too.

(*R3*) *Service Reputation:* The reputation system must provide reputation information about elementary services that have been consumed as a part of a composed service.

The recommendation process additionally rates alternative composition steps based on experience gained from previous composition processes. Reputation information about previously composed services is needed as feedback for the recommendation process in order to adapt its recommendation strategy by means of Reinforcement Learning. An OTF provider's experience, however, can be considered a business secret that must not be revealed to other market participants.

(*R4*) *Composed Service Reputation:* The reputation system must provide reputation information about composed services without revealing business secrets of OTF providers.

Users only interact with OTF providers and not with service providers directly (cf. Figure 1). As a consequence, a user's feedback mainly contains information about OTF providers and their composed services. Only once in a while may a user be able to additionally rate elementary services. For example, when using a composed service for an image processing task,

 ISBN: 978-1-61208-337-7

users may not be aware of all elementary services, e.g., of the filter service that reduces image noise. However, they may be able to rate an elementary service that implements an image compression algorithm, since the way the algorithm effects the execution result can be directly observed in terms of the size and quality of the generated image or video.

(*R5*) *Incomplete User Rating:* The reputation system has to consider that a user is most often only able to rate OTF providers and their composed services, while a user is only sometimes able to rate elementary services and never able to rate service providers.

B. Technical Requirements

The reputation system needs to provide access to the different reputation values mentioned in the previous section for the different parties illustrated in Figure 1. Those parties have diverse and variable needs for reputation value computations and access as well as interaction preferences. For the service recommendation process, recent ratings are more important to accelerate the learning process and therefore reputation values that put a higher weight on those ratings are desired (e.g., rather a geometric mean than an average with equal weights). In contrast, for a user, it might be preferable that a certain composed service has a very low failure rate and thus, during the provider selection process, reputation values that include historic values to a sufficient extent and put a higher weight on negative ratings have to be considered. The reputation system's functionality to process user feedback and to provide it as reputation information has to satisfy the diverse needs of the requesting parties.

(*R6*) *Flexible Feedback Processing:* The reputation system must support flexible processing of user feedback.

Certain restrictions may be applied: Concerning requirement (*R4*), reputation information about composed services shall be retrievable only by the OTF provider that originally accomplished the service composition process.

(*R7*) *Access Control:* The reputation system must implement access control to reputation values.

Furthermore, the reputation system shall support different interaction models. Parties, such as the OTF provider's service recommendation component, need new reputation information as soon as it is available. New reputation information has to be automatically forwarded by the reputation system without explicitly asking for it. Other processes that rarely need to retrieve reputation information, such as users or the service matching component, shall be able to access those data actively on demand to reduce the data traffic.

(*R8*) *Interaction:* The reputation system must support alternative interaction concepts. Reputation information must either be provided on demand triggered by a request event, or actively sent to a party as soon as new reputation information is available.

Furthermore, security and privacy protection are crucial issues—as we have already investigated more generally for the OTF Computing as well [2]. If users could arbitrarily rate any services (without having used them), the reputation system would not constitute any benefit. If any party would be able to manipulate the reputation values, users could not trust the provided values and thus the reputation system's benefit would be lost as well.

(*R9*) *Rating Authorization:* Only authorized users, i.e., users that performed a transaction with an OTF provider, are allowed to rate that transaction.

(*R10*) *Correctness:* The computed reputation value provided by the reputation system must be correct, i.e., it must not be possible for any party to manipulate the reputation value (computation).

Depending on the traded services on the market, users might only be willing to rate transactions if they can stay anonymous. They do not want to (publicly) reveal which services were consumed by them. It has been shown in the past that designing a reputation system that provides user anonymity is a challenging task [17].

(*R11*) *Anonymity of Rating User:* No party shall be able to relate (individual) ratings to users.

(*R12*) *Unlinkability of User Rating to Transaction:* The OTF provider must not be able to relate a rating to a transaction (previously executed with a certain user)—in order to achieve user anonymity.

IV. A Flexible Reputation System

This section introduces the conceptual design of our proposed solution in terms of a flexible reputation system. First, the system's internal processes as well as its interaction capabilities are described. Afterwards, we illustrate in particular how the system meets each requirement listed in Section III. An overview of our proposed solution is given in Figure 2. It shows the internal structure of our flexible reputation system as well as the interactions with the OTF Computing process. Both is further explained in the following.

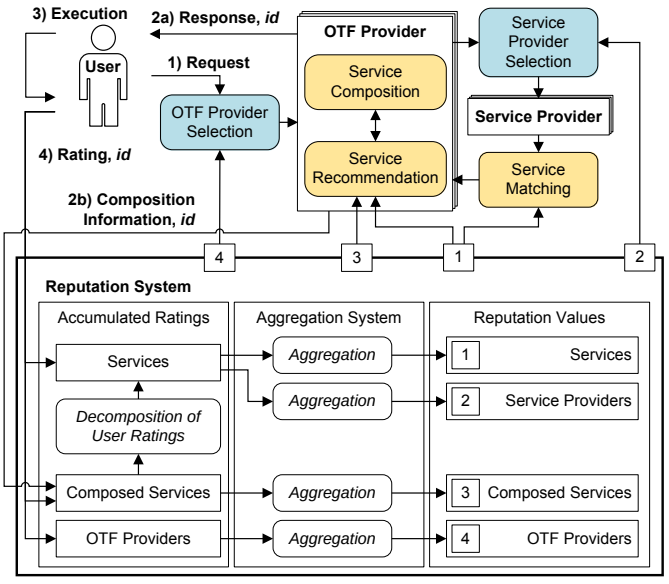


Figure 2: Proposed OTF Computing Reputation System. Internal structure and interactions with the OTF Computing process are depicted.

A. Basic Internal Structure

The reputation system is modeled as a stand-alone and independent component within the OTF Computing environment. The reputation values are derived by processing user ratings of services, composed services, as well as OTF providers. The internal structure can be divided into three main sections.

The *Accumulated Ratings* section provides functionality for accumulating raw values of incoming user ratings over time. To increase robustness, these values can be stored by means of a distributed storage system. The number of values to be stored is not necessarily restricted. However, depending on the available storage space and the amount of incoming values, outdated values may either be discarded or at least consolidated into a lower amount of values in the long run.

The *Aggregation System* provides functionality for processing a set of raw values in order to generate an aggregated representation. However, one can flexibly choose the set of raw values to be incorporated into the process, the actual aggregation function to be applied (e.g., arithmetic/geometric averaging, identifying the maximum or approximating the future trend by time series analysis) and the final representation (e.g., single scalars such as mean or median, or density functions in terms of their statistical parameters).

The *Reputation Values* section finally provides the interfaces for accessing the different reputation values of services, service providers, composed services, and OTF providers. When accessing reputation values, the set of raw user ratings to be considered, the actual aggregation function as well as the final representation can be flexibly specified. Reputation values are not stored within the system, but always computed on demand dependent on the previously mentioned specifications. This flexibility allows requests for reputation information to adapt to more complex reputation requirements imposed by users. For example, a user may want an image processing service with a reputation value higher than 4 based on at least 20 user ratings that are not older than 6 months. Another user may want an image processing service which has an average reputation value of 4, while no elementary service should have a reputation value less than 2.

B. Integration into the On-The-Fly Computing Process

Reputation values are consumed by the *Service Matching*, the *Service Recommendation*, the *Service Provider Selection*, and the *OTF Provider Selection* processes within the overall OTF Computing process. Beside flexibility regarding how a reputation value is internally computed, our proposed reputation system also provides flexible interaction capabilities. On the one hand, reputation values can be accessed by a *pull* approach whenever they are needed. Following this approach, the requester inherits the active role by asking for reputation data if and only if it is necessary. This solution is efficient when reputation information is needed less frequently (e.g., when a user wants to choose an OTF provider). On the other hand, a *push* approach shifts the active role to the reputation system. Reputation information is sent to a party as soon as new data is available. This approach also allows for creating a local cache of the latest reputation values without flooding the reputation system with redundant requests for possibly new information.

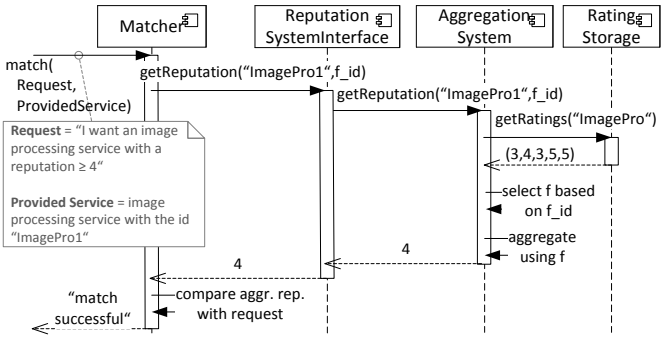


Figure 3: Simplified example interaction with the reputation system.

Figure 3 shows the interaction with the proposed reputation system using the example of the service matching process (matcher). During the OTF Computing process, the matcher is called for each elementary service that possibly satisfies an OTF provider's request (cf. Section II-B). In this context, Figure 3 illustrates the access of reputation information for exactly one elementary service by a *pull* approach.

The reputation matching process is initiated by providing the request information and the description of an elementary service and by calling the *match* operation. For the sake of simplicity, the request in the depicted example only shows an extract: An image processing service should have a minimum reputation value of 4. This request shall now be matched against an elementary service with id `ImagePro1`. The matcher asks the reputation system for a reputation value of service `ImagePro1` aggregated by means of an aggregation function with id `f_id`. Hence, the aggregation system fetches the relevant user rating values (3,4,3,5,5) from the storage, selects the corresponding aggregation function (here, arithmetic averaging), and computes an aggregated reputation value of 4. Based on this result, the matcher decides that the service matches to the request.

After a composed service was executed (*Step 3* in Figure 2), users are encouraged to provide feedback on their transactions. They are asked to rate composed services, OTF providers, and single services. The feedback in terms of user ratings is the foundation for generating reputation information within the reputation system. To be able to identify which composed service a rating belongs to, OTF providers attach an *id* to their response (*Step 2a* in Figure 2). This *id* corresponds to the particular structure of a composed service, meaning that identical composed services have identical *id*s. During the rating process for a composed service, this *id* is forwarded to the reputation system (*Step 4* in Figure 2).

Elementary services that are consumed as a part of a composed service cannot always be rated separately by the user. In fact, due to complex user requests, we expect that this is rarely possible. Thus, in order to still be able to provide reputation values for elementary services and to benefit from all information available, our reputation system decomposes user ratings of composed services. To enable this decomposition, the *id* the OTF provider sends with his response (cf. Figure 2) is reused: Simultaneously with his response to the user (*Step 2a* in Figure 2), the OTF provider sends the same *id* together with *composition information* to the reputation system (*Step*

 ISBN: 978-1-61208-337-7

2b in Figure 2).

As pointed out above, our reputation system for the OTF Computing shall provide flexibility, which also means that different implementations for the components are supported. We have already shown that such an implementation of a reputation system for the OTF Computing can be done in a *secure* and *privacy-preserving* way—respecting the requirements stated in Section III [18]. In contrast to related work, as covered in Section V, this approach only requires a single *reputation provider*, which is in line with the requirements of OTF Computing, and does not need any other components (such as a bulletin board). The approach is based on the Paillier cryptosystem [19] to provide a reputation value as an aggregation of individual user ratings without revealing anything about the individual ratings to any party.

C. Satisfying On-The-Fly Computing Requirements

Our proposed solution in terms of a flexible reputation system fulfills all requirements listed in Section III. This section points out how the reputation system fulfills each of these requirements in particular.

The proposed reputation system enables users to rate OTF providers, composed services, and—if possible—elementary services. Assured by the transfered *id*, in this context, only users that are involved in a particular transaction taking place on the OTF market, i.e., users that have requested, received and executed a particular composed service, are allowed to participate in the rating process. This ensures ratings by authorized users (*R9*). How to realize the rating process in particular (i.e., what kind of questions have to be asked and how a user rating value is represented) is beyond the scope of this paper.

Correctness of the provided reputation values is ensured by design. Reputation values are computed on demand by the system itself based on a pre-defined set of aggregation functions. Furthermore, the entire system is an independent component within the OTF Computing environment. As a consequence, manipulations of the computation process by other participants are eliminated (*R10*).

Anonymity of users (*R11*) as well as unlinkability of user ratings to transactions (*R12*) is ensured by the accumulation and aggregation functionality. For reasons of privacy protection, i.e., in order to not reveal individual user ratings, the reputation system always collects individual ratings and aggregates them. Although the single user ratings are stored within the reputation system, they are not accessible to market participants so that individual ratings are not traceable. In this context, it is important that the amount of accumulated user ratings is high enough and that the aggregation operation sufficiently condenses the user ratings such that it can be guaranteed that no information on individual ratings can be recovered. If not enough user ratings are included in the aggregation process (e.g., when not enough user ratings are available yet, or if a request explicitly specifies to only consider just a few user ratings), the reputation system will not provide a value but will raise an exception.

All processes that need reputation information within the entire OTF Computing process have access to the reputation system. The flexibility of our proposed solution enables each market participant to freely choose an interaction approach (*push* or *pull*) that is most appropriate with respect to the market participant's internal processes (*R8*). Furthermore, the process of generating reputation values can be adjusted by each market participant individually by specifying the set of user ratings to be considered, the actual aggregation function to be applied, and the final representation of the aggregated value (*R6*).

Reputation information about OTF providers (*R1*) is provided by the reputation system in a straight-forward manner. Users rate their satisfaction regarding the transaction with an OTF provider. These ratings are accumulated and aggregated by the reputation system and can be accessed by other users. The process of generating reputation information about composed services (*R4*) is similar. Users rate their satisfaction regarding the execution process and the execution result of a composed service. These ratings, again, are accumulated and aggregated by the reputation system. In comparison to the reputation of OTF providers, however, reputation information about composed services is OTF provider-related. In order to preserve business secrets, OTF providers can only access anonymized user ratings of composed services they originally sold (*R7*).

Besides being directly rated by users, ratings of elementary services also have to be derived from ratings for composed services (*R3*). For this purpose, OTF providers send information about their composed service to the reputation system. In order to not reveal their business secrets, this composition information, however, only consists of abstract, structural information. Only the set of elementary services included in a composed service is exposed, but not, for example, when and how often a particular service is called. This way, the provider's business secrets are protected, while it also allows for a mapping of the rating for a composed service to single services (*R5*).

Since users only interact with OTF providers, user ratings for service providers cannot be provided to the reputation system (*R5*). To overcome this problem, the aggregation system extrapolates from reputation information about elementary services to information about the associated service providers during the aggregation process (*R2*).

While composing services, reputation information about elementary services have most likely to be aggregated in order to choose composed services not only based on their (aggregated) non-functional properties, but also based on their overall reputation. How to determine this overall reputation, however, depends on the user requirements and the composition strategy of the respective OTF provider. If a user requires, e.g., all elementary services to satisfy a minimal reputation value, an OTF provider has to check the reputation value of each services individually. Another user might be satisfied with an average reputation value above a specific threshold. In this case, an OTF provider has to determine the average reputation value by aggregating all single values. Subsequently, the aggregated value and the threshold value have to be compared. In either case, aggregation of reputation values is not part of the reputation system itself. For that reason, a further investigation of how to appropriately integrate reputation information into service composition in addition to

 ISBN: 978-1-61208-337-7

common non-functional properties is beyond the scope of this paper.

V. Related Work

There is a lot of literature on reputation, both in economics and computer science. Our interpretation of reputation is used for instance by Shapiro [20] or as well by Bar-Isaac and Tadelis [21], who summarize the economic literature on reputation. Design aspects related to mathematically modeling a reputation system and challenges that arise with online transactions, are explicitly discussed by Friedman et al. [22] and Dellarocas [23], for example. More closely related, we identify three involved fields, *Reputation Systems*, *Privacy-Preserving Systems* and *Service Composition*, and their overlappings with each other as shown in Figure 4. In the following, we present related work which has been done within these overlappings in more detail.

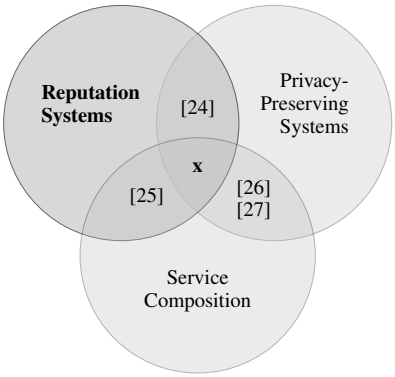


Figure 4: Overview of related work.

A. Reputation Systems and Privacy-Preserving Systems

Researchers have come up with *privacy-preserving* reputation systems in the past. Kerschbaum et al. [24] present a system which requires two centralized mutually mistrusting reputation providers in order to achieve anonymous user ratings. Users encrypt their ratings and send them to the first reputation provider which collects a number of ratings and then publishes them to a bulletin board. The second reputation provider retrieves the ratings from the bulletin board to decrypt and aggregate them before providing a (computed) reputation value. The approach is based on the Paillier cryptosystem [19]. However, the approach is too inflexible and complex to be used in our OTF Computing setting. We want to keep a lean OTF infrastructure with only one reputation provider and no other additional components, such as a bulletin board, used only by the reputation system.

B. Reputation Systems and Service Composition

Motallebi et al. [25] integrate *Component Reputation* and *Component Trust* in order to derive the reputation of a composed service from trust values for single services. They do this by taking into account the frequency of invocations of these services. However, this approach covers only some of our requirements for a reputation system in On-The-Fly Computing. For example, neither service providers are considered, nor is privacy or security a topic within their publication.

C. Service Composition and Privacy-Preserving Systems

Tbahriti et al. [26] identify privacy preservation as one of the most challenging problems in *Data-as-a-Service (DaaS)* services composition. DaaS is about combining web services for data publishing and sharing. In their proposed approach, *privacy policies* specify how collected data is treated and *privacy requirements* specify how the service-consuming services are expected to treat the provided data. Similarly, Costante et al. [27] come up with a solution for web service selection and composition that takes privacy into account. Users are able to specify their privacy preferences which are checked against the service providers' privacy policies. Only in the case of a successful match are the service providers' services selected and used for composition. Both approaches do not take into account reputation of elementary or composed services.

In contrast to related work, we pursue a *privacy-by-design* approach that builds privacy protection into the reputation system for OTF Computing. This allows us to prove that privacy is achieved rather than to rely on guarantees made by the participants.

D. Conclusion: Related Work

It is noteworthy that no work—to the best of our knowledge— that includes all the different fields mentioned above (the overlapping marked "x" in Figure 4) has been done. This is where we contribute with this paper: We are the first to present the requirements for such a system, describe a flexible solution, and point out further interesting research challenges that still need to be solved in the future.

VI. Research Challenges

The introduction of a reputation system in the OTF Computing in Section IV is conceptual and provides flexibility for further specifications. As research challenges, we highlight some of the trade-offs that result from the requirements imposed in Section III. A more detailed investigation of each of the research challenges is beyond the scope of this conceptual contribution and is planned to be considered in future work.

Efficiency in Learning versus Privacy Protection: The reinforcement learning approach, which is used to improve the service composition process, needs direct feedback after each composition. If the feedback is absent, the learning process is hampered. However, for reasons of privacy protection, no direct feedback is given to any party. Only an aggregated value of the accumulation of several individual ratings (feedback) is provided, as described in Section IV. Thus, a research challenge is to investigate the trade-off between privacy protection and learning efficiency. It has to be investigated how a delayed feedback after several service composition processes (accumulation) in an aggregated form affects the convergence behavior of the learning process.

Benefit of Privacy Protection: As discussed in this paper, the design of a privacy-preserving solution entails a multitude of trade-offs that need to be taken into account, e.g., the trade-off between privacy and learning mechanism efficiency. Thus, it needs to be investigated whether market participants are interested in implementing a privacy-preserving solution at all. We need to prove that privacy protection is a benefit

 ISBN: 978-1-61208-337-7

of OTF Computing and that users rather use such a market than any other which does not provide such strong privacy guarantees. Concerning the introduced reputation system, we want to examine whether users are more willingly providing ratings when their privacy is protected—which is not the case in any other state-of-the-art reputation system in use today.

Manipulation Resistance versus Privacy Protection: An important further issue is to obtain truthful user feedback. Ratings may be dishonest or randomly chosen [22]. So far we assumed that users have no incentives to strategically manipulate their feedback and moreover we supposed that feedback on a transaction is always provided. Truthful rating behavior is induced by *incentive compatible reputation mechanisms* [28] (and the references mentioned therein). To ensure privacy protection, several ratings need to be accumulated and aggregated. It has already been analyzed how the aggregation of ratings impacts the efficiency of a reputation mechanism [29] and how it influences incentives for truthful rating behavior [30]. An important next step now is to further understand the interplay of incentive compatibility and privacy protection. Therefore, a challenging question is whether and how it is possible to design reputation systems that induce truthful feedback and respect privacy protection.

Fuzzy Matching of Reputation Values: Another open issue is how reputation should be matched. Since the reputation of a service is not an objective measure, such as signatures or protocols, uncertainty might be introduced into the matching process. For example, as noted in our fuzzy matching survey [12], the user stating the request might tolerate variations (e.g., "I want a service with *approximately* five stars"), or the request might include requirements for which the corresponding information on the provider side do not exist yet (e.g., there has not been much feedback yet because the service is new on the market and thus the reputation is unclear). We are going to analyze how a fuzzy reputation matching can cope with such challenges.

Context-Specific Reputation: In our current system, we focus on the overall reputation of (composed) services and providers. However, in reality, reputation is rather context-specific [31]. For example, an image processing service could have a good reputation regarding the response time but a bad reputation regarding security. Thus, the reputation system should maintain vectors instead of single values for ratings and reputation. However, this also increases the complexity of the different components that access the reputation system. For example, matching could become much more detailed, but also less efficient in terms of performance.

Trust and Reputation: Another open issue is to distinguish between trust and reputation. Trust can be understood as a private reputation value in contrast to the public reputation value [32]. It needs to be analyzed whether reputation systems should distinguish between these concepts and how the whole scenario could benefit from it.

VII. Conclusion

In the context of OTF Computing, we use a reputation system to collect information about experiences users make with composed services in transactions. From an economic perspective, the buying decision of a user and the future sale opportunity of an OTF provider crucially depend on the current reputation value. Our contribution in this paper comprises the collection of requirements and the proposal of a conceptual solution for a flexible reputation system in OTF Computing. To fulfill the posed requirements, we identified necessary operations as well as additional properties and described their interaction. We analyzed the influence of reputation information on the processes and proposed the integration of a reputation system in the OTF Computing infrastructure. In our work, we put a special focus on composed services as well as on privacy. As part of our contribution, we combined approaches from the literature on reputation systems, service composition, and privacy protection. Finally, we presented research challenges that arise from conflicting objectives and deserve further investigations.

Acknowledgment

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

References

[1] M. Happe, F. M. auf der Heide, P. Kling, M. Platzner, and C. Plessl, "On-the-fly computing: A novel paradigm for individualized it services," in Proceedings of the Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS). IEEE, 2013.

[2] R. Petrlic, A. Jungmann, M. C. Platenius, W. Schäfer, and C. Sorge, "Security and Privacy Challenges in On-The-Fly Computing," in Tagungsband der 4. Konferenz Software-Technologien und -Prozesse (STeP 2014), 2014, to appear.

[3] "Collaborative Research Center 901 - On-The-Fly Computing," 2014, URL: http://sfb901.uni-paderborn.de [accessed: 2014-03-15].

[4] "Instagram," 2014, URL: http://www.instagram.com [accessed: 2014-03-15].

[5] V. T. Silva, R. Hermoso, and R. Centeno, "A Hybrid Reputation Model Based on the Use of Organizations," in Coordination, Organizations, Institutions and Norms in Agent Systems IV. Springer, 2009, pp. 111–125.

[6] J. S.P. Guedes, V. Torres da Silva, and C. Lucena, "A Reputation Model Based on Testimonies," in Agent-Oriented Information Systems IV. Springer, 2008, pp. 37–52.

[7] F. G. Mármol and M. Q. Kuhnen, "Reputation-based Web service orchestration in cloud computing: A survey," Concurrency and Computation: Practice and Experience, 2013.

[8] N. Hiratsuka, F. Ishikawa, and S. Honiden, "Service Selection with Combinational Use of Functionally-Equivalent Services," in Proceedings of the 18th IEEE International Conference on Web Services (ICWS), 2011, pp. 97–104.

[9] P. Bartalos and M. Bielikova, "Semantic Web Service Composition Framework Based on Parallel Processing," in Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC), 2009, pp. 495–498.

[10] M. Aiello, E. el Khoury, A. Lazovik, and P. Ratelband, "Optimal QoS-Aware Web Service Composition," in Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC), 2009, pp. 491–494.

[11] M. C. Platenius, "Fuzzy Service Matching in On-The-Fly Computing," in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE). ACM, 2013, pp. 715–718.

[12] M. C. Platenius, M. von Detten, S. Becker, W. Schäfer, and G. Engels, "A Survey of Fuzzy Service Matching Approaches in the Context of On-the-fly Computing," in Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering (CBSE). ACM, 2013, pp. 143–152.

[13] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press, 1998.

 ISBN: 978-1-61208-337-7

[14] A. Jungmann and B. Kleinjohann, "Learning Recommendation System for Automated Service Composition," in Proceedings of the 10th IEEE International Conference on Services Computing (SCC), 2013, pp. 97–104.

[15] A. Jungmann, B. Kleinjohann, and L. Kleinjohann, "Learning service recommendations," Int. J. Business Process Integration and Management, vol. 6, no. 4, 2013, pp. 284–297.

[16] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE), 2001, pp. 249–262.

[17] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in Proceedings of the IEEE Symposium on Security and Privacy (SP), 2008, pp. 111–125.

[18] R. Petrlic, S. Lutters, and C. Sorge, "Privacy-Preserving Reputation Management," in Proceedings of the 29th Symposium On Applied Computing. ACM, 2014.

[19] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Proceedings of the 17th international conference on Theory and application of cryptographic techniques. Springer, 1999, pp. 223–238.

[20] C. Shapiro, "Premiums for High Quality Products as Returns to Reputations," The Quarterly Journal of Economics, vol. 98, no. 4, 1983, pp. 659–680.

[21] H. Bar-Isaac and S. Tadelis, "Seller Reputation," Foundations and Trends in Microeconomics, vol. 4, no. 4, 2008, pp. 273–351.

[22] E. Friedman, P. Resnick, and R. Sami, "Manipulation-Resistant Reputation Systems," in Algorithmic Game Theory, Chapter 27. Cambridge University Press, 2007.

[23] C. Dellarocas, "Reputation Mechanism Design in Online Trading Environments with Pure Moral Hazard." Information Systems Research, vol. 16, no. 2, 2005, pp. 209–230.

[24] F. Kerschbaum, "A verifiable, centralized, coercion-free reputation system," in Proceedings of the 8th ACM workshop on Privacy in the electronic society (WPES), 2009, pp. 61–70.

[25] M. R. Motallebi, F. Ishikawa, and S. Honiden, "Component Trust for Web Service Compositions," in AAAI Spring Symposium Series, 2012.

[26] S.-E. Tbahriti, M. Mrissa, B. Medjahed, C. Ghedira, M. Barhamgi, and J. Fayn, "Privacy-Aware DaaS Services Composition," in Database and Expert Systems Applications, 2011, pp. 202–216.

[27] E. Costante, F. Paci, and N. Zannone, "Privacy-Aware Web Service Composition and Ranking," in Proceedings of the 20th IEEE International Conference on Web Services (ICWS), 2013, pp. 131–138.

[28] S. Phoomvuthisarn, "A Survey Study on Reputation-based Trust Mechanisms in Service-Oriented Computing," Journal of Information Science and Technology, vol. 2, no. 2, 2011, pp. 1–12.

[29] C. Dellarocas, "How Often Should Reputation Mechanisms Update a Trader's Reputation Profile?" Information Systems Research, vol. 17, no. 3, 2006, pp. 271–285.

[30] C. Aperjis and R. Johari, "Optimal Windows for Aggregating Ratings in Electronic Marketplaces," Management Science, vol. 56, no. 5, 2010, pp. 864–880.

[31] Y. Wang and J. Vassileva, "A Review on Trust and Reputation for Web Service Selection," in 27th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2007, pp. 25–25.

[32] R. Kiefhaber, G. Anders, F. Siefert, T. Ungerer, and W. Reif, "Confidence as a Means to Assess the Accuracy of Trust Values," in 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2012, pp. 690–697.

 ISBN: 978-1-61208-337-7

Performance Evaluation of OM4SPACE's Activity Service

Irina Astrova
Institute of Cybernetics
Tallinn University of Technology
Tallinn, Estonia
irina@cs.ioc.ee

Arne Koschel
Alexander Olbricht, Matthias Popp
Faculty IV, Department for Computer Science
University of Applied Sciences and Arts Hannover
Hannover, Germany
akoschel@acm.org

Marc Schaaf
Stella Gatziu Grivas
Institute for Information Systems
University of Applied Sciences Northwestern Switzerland
Olten, Switzerland
marc.schaaf@fhnw.ch

***Abstract*—OM4SPACE provides cloud-based event notification middleware. This middleware delivers a foundation for the development of scalable complex event processing applications. The middleware decouples the event notification from the applications themselves, by encapsulating this functionality into a component called Activity Service. This paper presents preliminary results of the performance evaluation for the Activity Service.**

***Keywords*—OM4SPACE; Activity Service; WebLogic JMS; Amazon SQS; Event-Driven Architecture (EDA); Service-Oriented Architecture (SOA); Complex Event Processing (CEP); cloud computing.**

I. INTRODUCTION

In 2010, the *University of Applied Sciences Northwestern Switzerland* in cooperation with the *University of Applied Sciences and Arts Hannover Germany* started a project called OM4SPACE [1]-[6]. The idea behind OM4SPACE was to merge Event-Driven Architecture (EDA), Service-Oriented Architecture (SOA), Complex Event Processing (CEP) and cloud computing together to provide cloud-based event notification middleware for *decoupled* communication between CEP application components on all the layers of a cloud stack, including infrastructures, platforms, components, business processes and presentations (see Figure 1). By decoupled, we mean that events are posted to the middleware without knowing if and how they are processed later.

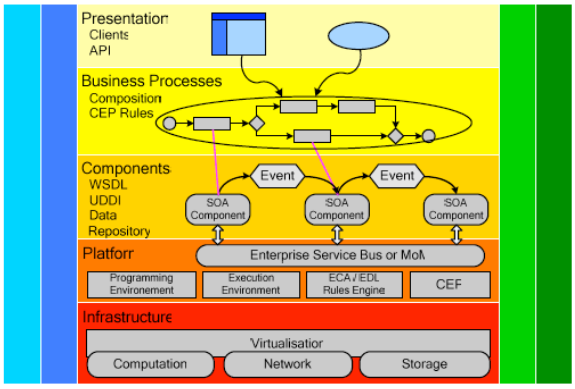


Figure 1. Cloud stack [3].

Performance is typically one of the top evaluation criteria for middleware products in general and OM4SPACE in particular. Since OM4SPACE is still relatively new, users expect that it will continue over time to improve its functionality, usability and reliability. However, users typically do want to get the best performance possible. Since the user's level satisfaction with OM4SPACE is largely determined by its performance, in this paper we evaluate the performance of OM4SPACE's Activity Service.

The rest of the paper is organized as follows. Section II presents the architecture of OM4SPACE. Section III describes the performance tests run against OM4SPACE. Section IV summarizes the results obtained during the performance tests and outlines future directions in the development of OM4SPACE.

II. ARCHITECTURE

Figure 2 gives an overview of the architecture of OM4SPACE, which includes the following components: *Event Producers* (also called *Event Sources*), *Event Consumers* and *Activity Service*.

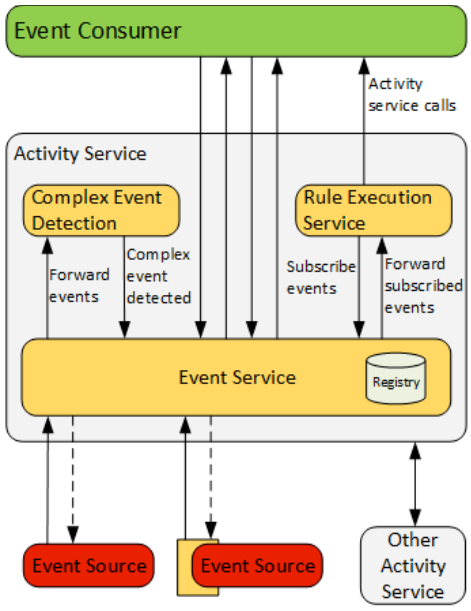


Figure 2. Architecture of OM4SPACE [3].

 ISBN: 978-1-61208-337-7

The Activity Service itself includes the following components:

- *Event Service:* This component receives events from Event Producers, pre-processes the events and delivers them to Event Consumers subscribed for those events. The Event Service contains a registry. Event Consumers look up events in the registry. If an Event Consumer finds an event of interest, it subscribes to that event.
- *Complex Event Detector:* This component receives events from the Event Service and derives from them new complex events, which are fed back into the Event Service for further processing.
- *Rule Execution Service:* This component receives events from the Event Service, evaluates them against CEP rules and triggers the rules into execution, which results in the execution of external action handlers that are provided by other third-party components.

The communication between all the components in the architecture is done through events, where an event is any kind of information sent as a *notification* from one component to another.

III. PERFORMANCE EVALUATION

One of the main advantages offered by OM4SPACE is its independence of channel service providers such as WebLogic, Amazon and Google because the Activity Service enables the transparent use of different transport technologies.

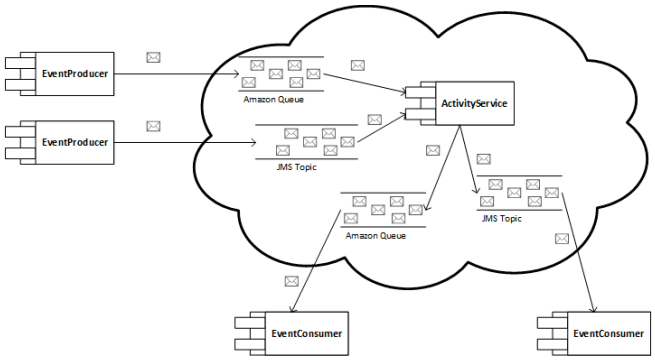


Figure 3. Transport technologies used by Activity Service.

In the current version of OM4SPACE, the Activity Service supports the following transport technologies:

- WebLogic JMS, which serves as an example of a *topic* service.
- Amazon SQS, which serves as an example of a *queue* service.

Once an Event Producer has sent events to the channel, the Activity Service located in a public cloud will forward the events to the channel of an Event Consumer that is subscribed for those events (see Figure 3). A decision on which channel to use for sending events is left solely to the Event Producer. Similarly, a decision on which channel to use for receiving events is left solely to the Event Consumer. For example, the Event Producer can select a JMS topic because it is not chargeable, whereas the Event Consumer can select an SQS queue because it is highly available (i.e., the availability of an SQS queue is not affected if the cloud instance fails).

A. Tests

We conducted the performance evaluation to answer the following questions:

- Will the Activity Service (sitting between the Event Producer and the Event Consumer) affect the time needed for events to reach their destination?
- If it does, will the performance still be good?

The answers to these questions were important because the application areas for OM4SPACE include smart grids [6] that need to address the challenges related to the constantly increasing number of events and near real-time reaction on those events.

To answer the questions above, we performed the following tests:

- *T1*: The Activity Service was not used. Events were sent via a JMS topic and received via the same topic.
- *T2*: The Activity Service was used. Events were sent via a JMS topic and received via another JMS topic.
- *T3*: The Activity Service was not used. Events were sent via an SQS queue and received via the same queue.
- *T4*: The Activity Service was used. Events were sent via an SQS queue and received via another SQS queue.
- *T5*: The Activity Service was used. Events were sent via a JMS topic but received via an SQS queue.
- *T6*: The Activity Service was used. Events were sent via an SQS queue but received via a JMS topic.

These tests were intended to prove or disprove the following hypotheses:

- *H1*: JMS alone can achieve better performance than JMS interconnected with the Activity Service.
- *H2*: SQS alone can achieve better performance than SQS interconnected with the Activity Service.
- *H3*: There can be a difference in the performance of JMS alone and SQS alone.
- *H4*: There can be a difference in the performance of JMS interconnected with the Activity Service and SQS interconnected with the Activity Service. This difference can be the same as above.
- *H5*: The number of events can affect the performance of JMS alone.
- *H6*: The number of events can affect the performance of SQS alone.
- *H7*: The number of events can affect the performance of JMS interconnected with the Activity Service.
- *H8*: The number of events can affect the performance of SQS interconnected with the Activity Service.

We performed the tests in the following way:

 ISBN: 978-1-61208-337-7

- Each test was executed with a different number of events (100, 500 and 1000) to see how the event number affects the performance.
- Each test was executed ten times to calculate the average where outliers were still visible.
- In each test, the time from sending the first event to receiving the last one was measured using a Java method `System.currentTimeMillis` (which returns the current time in msecs).
- Depending on the test, either all the components (the Event Producer, the Activity Service and the Event Consumer) were running on the same cloud instance or each component was running on its own cloud instance. Because of the decision to use SQS, Amazon EC2 was used as the cloud. Generally, Event Producers and Event Consumers are not limited to the components of a public cloud where the Activity Service is located. Rather, they can be located in private clouds or in some other public clouds (see Figure 6).

The measurements were made with two Ubuntu Linux 9.10 systems, which both used Sun Java 1.6.0. The machine, which hosted the Event Producer, the Activity Service and the Event Consumer, was a dual core system with 4GB memory. The machine for the cloud was a quad core system with 8GB memory. The two machines were interconnected with a gigabit Ethernet.

B. *Test Results*

The test results proved H1, H2, H3, H5, H6, H7 and H8, but disproved to some degree H4.

Figure 4 summarizes the test results for T1 and T2. What attracts our attention is a very good performance that JMS demonstrated in all the tests. For example, sending and receiving 100 events via JMS interconnected with the Activity Service took only 1286 msecs. But as one could expect, this time was longer than without the Activity Service.

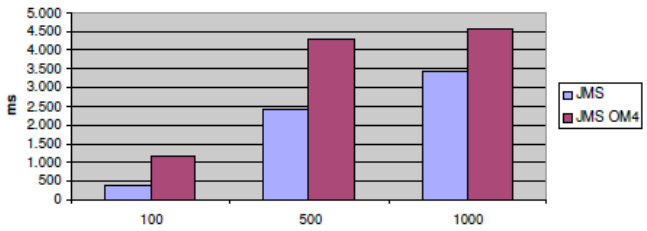


Figure 4. Sending and receiving 100, 500 and 1000 events: JMS alone vs. JMS interconnected to Activity Service.

One could expect that the time would increase with an increase of the number of events. Indeed, for sending and receiving 500 events, JMS interconnected with the Activity Service needed 3184 msecs more than for sending and receiving 100 events. However, of peculiar interest is the fact that for sending and receiving 1000 events, JMS interconnected with the Activity Service needed only 305 msecs more than for sending and receiving 500 events. In both cases, the average time was about 4500 msecs. Therefore, we suggest that extra time needed for sending and receiving 100 events was the time that the Activity Service needed for initialization.

The left column in Table I shows the time needed for JMS to send and receive 500 events without the Activity Service, whereas the right column with the Activity Service. What attracts our attention is the sharp deviation in the ten test runs in both cases. For example, the time needed for sending and receiving 500 events via JMS interconnected with the Activity Service was between 3332 and 6300 msecs (i.e., the test results differed in almost two times).

TABLE I. SENDING AND RECEIVING 500 EVENTS: JMS ALONE VS. JMS INTERCONNECTED TO ACTIVITY SERVICE

JMS	JMS OM4
851	6300
836	3942
3956	3484
3895	4247
1525	3323
713	4522
3865	3360
3835	5247
4023	4168
887	4258
2439	**4285**

Figure 5 summarizes the test results for T3 and T4. What attracts our attention is that SQS alone was much slower than JMS alone – in fact, it was even slower than JMS interconnected with the Activity Service. For example, sending and receiving 100 events via SQS already took 13,412 msecs. With the Activity Service interconnected, that time was even longer (viz., 373,678 msecs). However, as one could expect, the time increased with an increase of the number of events but quickly, especially when SQS was interconnected with the Activity Service.

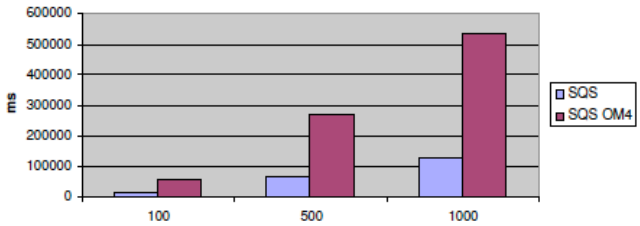


Figure 5. Sending and receiving 100, 500 and 1000 events: SQS alone vs. SQS interconnected to Activity Service.

Our tests showed that SQS alone was up to 36 times slower than JMS alone. One could expect that the same would keep true if the Activity Service were used. In fact, SQS interconnected with the Activity Service was up to 120 times slower than JMS interconnected with the Activity Service. Therefore, we suggest that the Activity Service greatly affected the performance, when SQS was used as the transport technology.

The left column in Table II shows the time needed for SQS to send and receive 500 events without the Activity Service, whereas the right column with the Activity Service.

 ISBN: 978-1-61208-337-7

Although the time was extremely long, it was almost constant for all the ten test runs (viz., between 262,867 and 270,603 msecs for sending and receiving 500 events) when SQS was interconnected with the Activity Service.

TABLE II. SENDING AND RECEIVING 500 EVENTS: SQS ALONE VS. SQS INTERCONNECTED TO ACTIVITY SERVICE

SQS	SQS OM4
65206	265602
65489	264818
66067	270338
64678	264498
67736	264092
65099	270603
64350	266591
65396	266645
65240	268499
64476	262867
65374	**266455**

While executing the tests, we noticed that the Activity Service demonstrated the worst performance when events were sent via an SQS queue and received via another SQS queue (T4). The performance improved when events were sent via a JMS topic but received via an SQS queue (T5). The performance became even better when events were sent via an SQS queue but received via a JMS topic (T6). Therefore, we suggest that sending events via an SQS queue does not take extra time but receiving events does. That is, the problem is that when the Activity Service deposits events to an SQS queue, the Event Consumer receives them with a big delay. Therefore, the performance problem might be resolved by optimizing the way the Activity Service works or with better implementation of the source code (which is written in Java).

IV. CONCLUSION AND FUTURE WORK

The performance of the Activity Service was evaluated. Our tests showed that sending and receiving events via JMS interconnected with the Activity Service took up to three times longer than without the Activity Service. However, that time was still short and increased slowly with an increase of the number of events. Therefore, we consider the performance to be very good, when JMS is used as the transport technology.

By contrast, the use of SQS could cause a performance bottleneck. Our tests showed that SQS itself was up to 36 times slower than JMS. (This was probably due to the distributed nature of an SQS queue). But with the Activity Service interconnected, the time for sending and receiving events increased up to 20 times more, resulting in almost 330,000 msecs delay.

Since OM4SPACE is relatively new, it will continue over time to improve its performance. In addition, OM4SPACE seeks to support more transport technologies, including Google App Engine and WS Notification. Therefore, in the future, we intend to execute more performance tests in order to obtain new test results.

ACKNOWLEDGMENT

Irina Astrova's work was supported by the Estonian Centre of Excellence in Computer Science (EXCS) funded mainly by the European Regional Development Fund (ERDF). Irina Astrova's work was also supported by the Estonian Ministry of Education and Research target-financed research theme no. 0140007s12.

REFERENCES

[1] M. Schaaf, A. Koschel, and S. G. Grivas, "Event processing in the cloud environment with well-defined semantics," The 1st International Conference on Cloud Computing and Services Science (CLOSER 2011), May 2011, pp. 176-179.

[2] A. Koschel, M. Schaaf, S. G. Grivas, and I. Astrova, "An ADBMS-style Activity Service for cloud environments," The 1st International Conference on Cloud Computing, GRIDs and Virtualization (CLOUD COMPUTING 2010) IARIA, Nov. 2010, pp. 80-85.

[3] R. Sauter, A. Stratz, S. G. Grivas, M. Schaaf, and A. Koschel, "Defining events as a foundation of an event notification middleware for the cloud ecosystem," The 15th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2011), Sep. 2011, LNCS, vol. 6882, pp. 275-284, doi:10.1007/978-3-642-23863-5_28.

[4] M. Schaaf, A. Koschel, and S. G. Grivas, "Towards a semantic definition for a cloud-based event notification service," The 3rd International Conference on Cloud Computing and Services Science (CLOSER 2013), May 2013, pp. 345-349.

[5] I. Astrova, A. Koschel, L. Renners, T. Rossow, and M. Schaaf, "Integrating structured peer-to-peer networks into OM4SPACE project," The 27th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013), Mar. 2013, pp. 1211-1216, doi:10.1109/WAINA.2013.88.

[6] A. Koschel, A. Hödicke, M. Schaaf, and S. G. Grivas, "Supporting smart grids with a cloud-enabled Activity Service," The 27th International Conference on Informatics for Environmental Protection (EnviroInfo 2013), Sep. 2013, pp. 205-213.

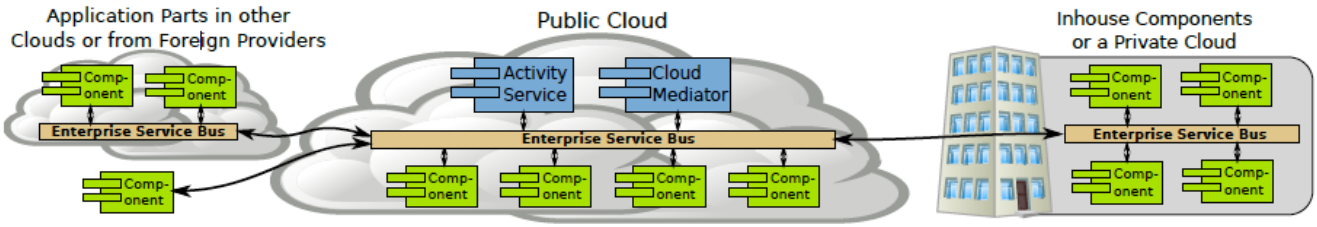


Figure 6. Distribution of OM4SPACE components [2].

Evaluating the Data Quality and the Uncertainty in Electroencephalogram Signals for a Neuromarketing Service which Computes Attentional Engagement

Wuon-Shik Kim, Sang-Tae Lee
Korea Research Institute of Standards and Science
Daejeon, Republic of Korea
wskim@kriss.re.kr, stlee@kriss.re.kr

Yaeeun Kim, Hyoung-Min Choi
PHYSIONICS Co., Ltd.
Daejeon, Republic of Korea
rosacalla@kaist.ac.kr, Hmchoi78@gmail.com

Abstract— **Objective and quantitative data, which indicate when and how much moviegoers are engaged with movies is important for movie makers when creating a film. However, with the traditional method of a review questionnaire, it is difficult to determine precisely the degree to which moviegoers were engaged and when this occurred. To evaluate the Attentional Engagement (AE) precisely, we used electroencephalogram (EEG) on Japanese students who were watching the American movie *Iron Man*. We found a significant decrease in the EEG power in the low-alpha-frequency band while the participants watched film content evoking subjective higher attention. To use our results as reference data for a neuromarketing service, we suggest a procedure with criteria evaluating the quality level of data. According to this procedure, the EEG power values of AE for the movie *Iron Man* can serve as standard reference data with quality level of validated.**

Keywords-neuromarketing service; electroencephalogram; data quality; uncertainty; attentional engagement.

I. INTRODUCTION

Immersion, like flow, has been used to describe the degree of an experience of feeling deeply engaged with types of media such as novels, movies, computer games, and virtual reality. To enter engagement, the first stage of immersion, users (e.g., gamers, moviegoers, etc.) have to invest time, effort, and attention [1][2]. Objective and quantitative measurements are important for media makers to determine when and how much media users are engaged with media. Standard marketing techniques employed for movies thus far have involved the use of interviews and questionnaires after participants view a movie. While useful, it is difficult with these methods to determine precisely the degree to which they were engaged or when they became engaged. If technology, which can measure variations in media users' levels of engagement is developed, it would help media makers when designing media content to induce the intended level of engagement. Currently, the focusing of attention can be monitored by measuring associated changes in brain electrical activity by means of electroencephalogram (EEG) [3][4]. Thus, in principle, EEG measures have the potential to provide a more direct and objective method for gauging the intensity and nature of moviegoer engagement [5][6].

Recently, with neuromarketing technology, defined as the application of neuroscientific methods to analyze and understand human behavior in relation to markets and marketing exchanges, marketing-relevant human behaviour can be understood [7][8][9][10][11]. However, it is difficult to measure physiological signals compared to the ease with which subjective review questions can be given. Therefore, many researchers want to share physiological signal data. Hence, a neuromarketing service, which computes the Attentional Engagement (AE) of moviegoers by measuring physiological signals is necessary and important in film production. In addition, to ensure the reliability of the service, it is also important to evaluate the quality of the data. The issues and practices with regard to a data evaluation include accessibility in data collection, reproducibility of basic evaluations, consistency in relational analyses, and predictability in modelling [12]. The purpose of a measurement is to determine the value of the measurand, that is, the value of the particular quantity to be measured. In general, the result of a measurement is only an approximation or estimate of the value of the measurand and thus is complete only when accompanied by a statement of the uncertainty of that estimate. The uncertainty in the result of a measurement generally consists of several components which may be grouped into two categories according to the way in which their numerical value is estimated. These are A) those which are evaluated by statistical methods, and B) those which are evaluated by other means [13].

In the present study, to serve as a reference data for neuromarketing in film industry, we measured the AE of participants as they watched movies. To validate the reliability of the reference data, we suggested a procedure with criteria, which evaluates the quality level of reference data. Finally, we applied the procedure to the present reference data. The remainder of this paper is organized as following. Section II describes the experimental method including participants, movie and audio-video system as stimulus, experimental procedure, questionnaire, EEG recording, statistical analysis, and evaluating the data quality. Section III describes the results including self-report measure, the EEG response, evaluating the uncertainty, and accrediting the level of data quality. Conclusion and future work are discussed in Section IV.

II. METHOD

A. Participants

The participants recruited were 12 right-handed students from the University of Tsukuba (UT) in Japan. Potential participants were excluded if they reported any history of neurological problems, took within 6 hours caffeine or any drug related to arousal, and had already seen the movie *Iron*

 ISBN: 978-1-61208-337-7

Man. Due to the approximate time of two hours required for the long experiment, measurement device errors (n=1), drowsiness (n=1) and severe eye blink noise (n=2) caused the data for some of the participants and related measures to be unavailable for use here. Therefore, only eight participants (7 females; mean age: 22.3 years, range 20-28 years) were analyzed to evaluate AE.

B. Stimulus

To avoid previous movie experience by the Japanese participants as much as possible, we played the American movie *Iron Man* (manufacturer: Marvel, year produced: 2008, running time: 2 hours 50 seconds), which was considered to be a typical American 'superhero' movie. It was played using a PC-based beam projector system (Epson EB series) with a 5.1 surround-sound system (Cambridge DTT-3500). The main story and the time information for Film segment 1 and 9 are shown in Table I, and the typical scenes are in Fig. 1 (refer to "*D. Self-report measure*s").

TABLE I. CONTENT STORY OF *IRON MAN*

Film ID[a]	Main story of each film content segment	Start (s)	End (s)
1	Tony Stark (the main character in the film) and his colleagues were telling indecent jokes.	50	147
9	Tony Stark was flying through the night sky with the successfully completed *Iron Man* suit.	3,598	3,720

a. Film ID is the segment number of film content

Figure 1. Typical scenes selected from *Iron Man*.

C. Experimental Procedure

Upon arrival, the participants were fully informed of the purpose of the experiment and its procedure, and all signed a consent form that was approved by the Institutional Review Board (IRB) of UT. They were then led to a small sound-attenuated room equipped with a 150-inch wide screen. The procedure of the experiment consisted of three sessions. The first session (baseline session) was done to measure the physiological signals of the participants during a five-minute baseline state. The second session (movie session) sought to measure physiological signals from participants while they watched the movie, and this session took approximately two hours. The third session (questionnaire session), which took place after the movie, evaluated how the participants felt while watching the movie.

D. Self-report measures

With the help of movie narrative and storytelling experts, 11 film-content segments were selected as relatively meaningful parts of the story from the full movie. After watching *Iron Man*, the participants took part in a questionnaire session. For each of the 11 film-content segments, the participants were asked to rate the affection and AE that they felt while watching the movie on a Likert scale ranging from 1 to 9. The subjective questionnaire for affection was prepared by adapting a questionnaire from the model devised by Russel [14]. Finally, we selected two film-content segments, one with the highest score and one with the lowest score, on subjective AE.

E. EEG recording and data analysis

Two-channel EEG signals (Fp1 and Fp2 according to the 10-20 system) for each of the three participants were recorded during the movie session using an MP150 system and AcqKnowledge software version 4.2 (Biopac, USA). To synchronize the physiological signals with the corresponding scene, we measured the changes in the luminance in the scenes with a photometer. By measuring the rapid changes in the levels of luminance, 35 s from the beginning of *Iron Man* was determined as the reference time at which to synchronize with the starting point of the EEG. We collected the EEG activity with a personal computer at a sampling rate of 1,000 Hz. The EEG traces were analyzed in one-second intervals with a step of 0.5 s (50% overlapped). The fast Fourier transform was then computed on 50%-overlapped groups of 1,000-sample Hanning windows for all artifact-reduced data segments to obtain the Power Spectral Density (PSD) for each segment in the approximately two-hour film. Next, the EEG power in the low-beta-frequency band (13-20 Hz) was calculated from the PSD for each one-second segment. EEG signals were analyzed for two film-content segments, i.e., those with the highest and lowest AE scores as determined from self-reports. To examine the changes in the EEG signals while the participants watched these two film-content segments, we selected two intervals in each film-content segment. The duration of each interval was set to 60 s. The mean values of each EEG signals were calculated over two 30-second overlapped 60-second-long intervals for the two film-content segments separately. To analyze the EEG signals, MATLAB S/W version 7 was used.

F. Statistical Analysis

For the two film-content segments selected corresponding to the highest and the lowest subjective AE scores, a paired-samples t-test was carried out for the self-report ratings (AE and affection). The EEG data were subjected to a two-way analysis of variance, with Film Content (the content segments with the lowest AE and the highest AE) and Interval (Interval 1: 0–60 s and Interval 2: 30–90 s from the beginning of each film-content segment) as repeated-measures factors. The eta-squared statistic (η^2), indicating the proportion between the variance explained by one experimental factor and the total variance, is reported. Statistical analysis was carried out using SPSS ver. 21.

G. Evaluating Data Quality

To ensure the reliability of the physiological data, which will be serviced for neuromarketing in movie industry, we established a procedure evaluating the credibility of the data (Table II).

 ISBN: 978-1-61208-337-7

TABLE II. PROCEDURE USED TO EVALUATE THE DATA QUALITY

Standard Reference Data as validated (1 ~ 8)
1. Specification of quantity to be measured
Stimulus, Target users (Participants), Physiological signals, Measurand
2. Measurement method and procedure
Measurement method, Measurement procedure, Measurement index
3. Theoretical basis of the measurement index
4. Control of factor influencing to measurement
5. Uncertainty of measurement method
6. Uncertainty of values measured
7. Reproducibility
8. Consistency
Standard Reference Data as verified (9)
9. Predictability based on modelling
Standard Reference Data as Certified (10)
10. Overall evaluation by two specialists

This procedure includes multiple acceptance categories with differing acceptance criteria. If the data satisfy criteria items from 1 to 8, then it is qualified as Standard Reference Data (SRD) of validated: satisfy from 1 to 9, then qualified as SRD of verified: satisfy from 1 to 10, then qualified as certified.

III. RESULTS AND DISCUSSIONS

A. Self-report

As Fig. 2 illustrates, the participants reported the highest AE for Film Content 9 (7.75±0.97) and the lowest AE for Film Content 1 (5.21±1.16). The self-report ratings for AE ($t=-7.61$, $p<0.001$), valence ($t=-4.88$, $p<0.001$), and arousal ($t=-4.50$, $p<0.01$) were significantly higher in Film Content 9 compared to those in Film Content 1.

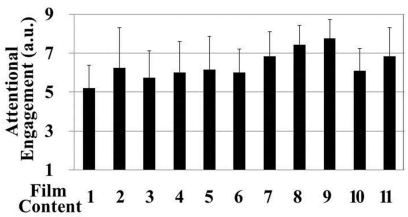


Figure 2. Self-report ratings of attentional engagement for 11 film-content segments in the movie *Iron Man*.

B. EEG Responses

To investigate the changes of the EEG signals while the participants watched Film Contents 1 and 9, we defined Interval 1 as being from 50 to 110 s for Film Content 1 and from 3,598 to 3,658 s for Film Content 9 and Interval 2 as being from 80 to 140 s for Film Content 1 and from 3,628 to 3,688 s for Film Content 9. We defined an attention index (EEG-attention) as the sum of the EEG powers in the low-beta-frequency band at Fp1 and at Fp2 because the low-beta-frequency band is known to be activated during attention. For EEG-attention, the main effect of the Interval was significant ($F(1, 11)=5.23$, $p<.05$, $\eta^2=0.37$), indicating that EEG-attention is higher in the second interval as compared to the first in the reaction to both Film Content 1 and Film Content 9. However, neither the main effect of the Film Content nor the Film Content by Interval interaction was significant. EEG-attention increased from 32.67±23.45 to 36.96±23.75 [ms^2] ($t=-2.06$, $p=0.069$), which was nearly significantly in the reaction to Film Content 9, and from 31.36±20.58 to 31.54±19.53 [ms^2] ($t=-1.44$, $p>0.05$) in the reaction to Film Content 1 (Fig. 3).

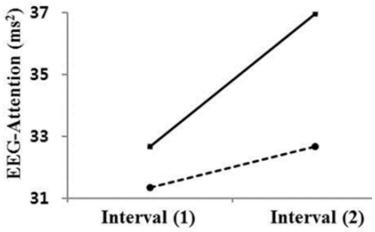


Figure 3. Attention index of the EEG (EEG-Attention). Dotted line denotes Film Content 1 and continuous line denotes Film content 9.

C. EvaluatingUncertainty

The Type A uncertainty of EEG power $u_A(P)$ for Interval 1 of Content segment 1 was calculated from (1). In the same way, the Type A uncertainty of EEG power $u_A(P)$ was calculated for each interval of Content segment 1 and 9, respectively, and summarized in Table III. The Type B uncertainty of EEG potential $u_B(V)$ is the sum of the uncertainty when amplifying the EEG potential using a voltage amplifier (2): $u_{Amp}(V)$, when digitalizing from the analog output of the EEG amplifier with a voltage range of 1 mV and with the quantization resolution B being 16 (3): $u_q(V)$, and when the fast Fourier transformation with a sampling frequency f_s of 1,000 (4): $u_{FFT}(V)$ [15]. The $u_{Amp}(V)$ was calculated from the expanded uncertainty of voltage amplifier $U_{Amp}(V)$, 0.042 V, which was calibrated by Korean national standard. The Type B uncertainty of EEG potential $u_B(V)$ was calculated from (5). The Type B uncertainty of EEG power $u_B(P)$ for Interval 1 of Content segment 1 was calculated from (6). The combined standard uncertainty of EEG power $u_C(P)$ for Interval 1 of Content segment 1 was calculated from (7). In the same way, the Type B uncertainty of EEG power $u_B(P)$ and the combined standard uncertainty of EEG power $u_C(P)$ were calculated for each interval of Content segment 1 and 9, respectively, and summarized in Table IV. The expanded uncertainty $U(P)$ of EEG power with coverage factor k being 2 for Interval 1 of Content segment 1 was calculated from (8). In the same way, the expanded uncertainty $U(P)$ of EEG power was calculated for each interval of Content segment 1 and 9, respectively, and summarized in Table IV. Finally, the EEG power values with expanded uncertainty are summarized in Table V.

 ISBN: 978-1-61208-337-7

$$u_A(P) = \frac{s}{\sqrt{n}} = 7.07\ \mu V^2, \quad s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}} \qquad (1)$$

$$u_{Am\ p}(V) = \frac{0.042}{2} = 0.021\ \mu V \qquad (2)$$

$$u_q(V) = \frac{a}{\sqrt{3}} = V_{range} \times \frac{2^{-B}}{2\sqrt{3}} = 1\ m\,V \times \frac{2^{-16}}{2\sqrt{3}} = 0.004\ \mu V \qquad (3)$$

$$u_{FFT}(V) = \sqrt{\frac{u_q^2}{2f_s}} = \sqrt{\frac{0.004^{\,2}}{2\times 1000}} = 9.84 \times 10^{-5} \mu V \qquad (4)$$

$$u_B(V) = \sqrt{u_{Am\ p}^2 + u_q^2 + u_{FFT}^2} = 0.021\ \mu V \qquad (5)$$

$$u_B(P) = P\sqrt{\left(\frac{2u_B(V)}{V}\right)^2} = 2 \times 36.34\ \mu V^2 \times \frac{0.021\ \mu V}{5.79\ \mu V} = 0.27\ \mu V^2 \qquad (6)$$

$$u_C(P) = \sqrt{u_A^2(P) + u_B^2(P)} = 7.08\ \mu V^2 \qquad (7)$$

$$U(P) = k\,u_C(P) = 14.16\ \mu V^2,\ k = 2 \qquad (8)$$

TABLE III. THE TYPE A UNCERTAINTY OF THE EEG POWER: $u_A(P)$

Participant ID	EEG Power values and Type A uncertainty [μV^2]			
	Content segment 1		*Content segment 9*	
	Interval 1	*Interval 2*	*Interval 1*	*Interval 2*
1	42.82	42.49	50.32	51.81
2	22.29	22.76	23.73	26.47
3	59.39	65.36	62.71	59.55
4	21.00	21.02	30.57	30.91
5	13.13	17.57	17.13	19.35
6	18.13	19.73	19.78	21.11
7	63.06	58.54	76.99	87.91
8	50.91	43.52	30.37	37.67
Average	36.34	36.37	38.95	41.85
Uncertainty Type A: $u_A(P)$	7.07	6.64	7.75	8.29

TABLE IV. THE EXPANDED UNCERTAINTY OF THE EEG POWER: $U(P)$

Items	Type B, combined, and expanded uncertainty for EEG power [μV^2]			
Segments in Film-content	*Content segment 1*		*Content segment 9*	
Intervals in Segments	*Interval 1*	*Interval 2*	*Interval 1*	*Interval 2*
Uncertainty Type B: $u_B(P)$	0.27	0.27	0.27	0.27
Uncertainty Combined: $u_C(P)$	7.08	6.65	7.75	8.29
Uncertainty Expanded: $U(P)$	14.16	13.30	15.50	16.58

TABLE V. THE EEG POWER VALUES WITH EXPANDED UNCERTAINTY

EEG Power value with uncertainty for attentional engagement [μV^2]			
Content segment 1		**Content segment 9**	
Interval 1	**Interval 2**	**Interval 1**	**Interval 2**
36.34 ± 14.16	36.37 ± 13.30	38.95 ± 15.50	41.85 ± 16.58

D. Accrediting the level of Data Quality

To assure the quality of the data for neuromarketing service, we evaluated the level of the reliability according to the procedure evaluating data quality (Table I). Because the EEG power values with the uncertainty from present study satisfy criteria items from 1 to 8 in Table VI, it can be served as SRD with quality level of validated.

TABLE VI. PROCEDURE USED TO EVALUATE THE QUALITY OF PRESENT DATA

Standard Reference Data as Validated (1 ~ 8)
1. Specification of quantity to be measured
Stimulus
- Genre/ Title: superhero film/ *Iron Man* - Producer/ Produced year: Marvel/ 2008 - Running time: 2 hours 50 seconds
Target users (Participants)
- Nationality: Japan - Gender/ Age/ Number of samples: 1 male (age: 20 years), 7 females (mean age: 22.3 years, range: 20-28 years)
Measurand observable: potential of EEG
Measurand to know: power of EEG
2. Specification of measurement method and procedure
Measurement method: EEG at Fp1 and Fp2 in 10-20 system
Measurement procedure: consists of three sessions (baseline session, movie session, questionnaire session)
Measurement index: EEG power in low-beta-frequency band for attentional engagement
3. Theoretical basis of the measurement index
EEG power in low-beta-frequency band is known to be activated during attention.
4. Control of factor influencing to measurement
To reduce noise, the impedance between electrodes was kept below 5 kΩ.
5. Traceability to national standard
EEG amplifier was calibrated by Korean national standard.
6. Evaluating uncertainty
Combined uncertainty of Type A and Type B was evaluated for EEG power.
7. Reproducibility
Reproducibility was satisfied during preliminary experiment.
8. Consistency
EEG index of attentional engagement consistent with the subjective self-report.
Standard Reference Data as Verified (9)
9. Predictability based on modelling
Modelling will be carried out as future work.
Standard Reference Data as Certified (10)
10. Overall evaluation by two specialists
Overall evaluation will be carried out as future work.

 ISBN: 978-1-61208-337-7

In this study, we suggested an evaluation method with procedure to assure the reliability of the neuromarketing service data. As validated in credibility according to the procedure evaluating data quality, the present data as reference standard can be serviced to neuromarketing in movie industry. The data include as follows: information of stimulus and participants, raw signals of EEG, and EEG power in low-beta-frequency band for AE with traceable uncertainty.

IV. CONCLUSION AND FUTURE WORK

The present study sought to suggest an evaluation method, which will be used to assure the reliability of a physiological data for neuromarketing service. In conclusion, the method evaluating the reliability of EEG data from participants while watching American movie *Iron Man* results in successful application. The EEG power values with the uncertainty, which is traceable to a Korean national standard can be used as validated SRD for a neuromarketing service. This data can be serviced as reference standard to compute AE of moviegoers while watching the American movie *Iron Man.*

In the future, to enhance the reliability level of present data for neuromarketing service, it will be necessary to construct a model, which predicts the AE of participants while watching movies.

ACKNOWLEDGMENT

The authors thank the students of the University of Tsukuba who participated and assisted in this study, and especially Prof. SeungHee Lee and her students in the Department of Science of Kansei Design. This research was supported by two projects. The one is 'A Technical Development of the Global Code Based on the Story,' sponsored by the Korea Creative Content Agency and the Ministry of Culture, Sports and Tourism. The other is 'The Development and the Dissemination of National Standard Reference Data,' sponsored by Korean Agency for Technology and Standards.

REFERENCES

[1] H. Qin, P. P. Rau, and G. Salvendy, "Measuring Player Immersion in the Computer Game Narrative," International Journal of Human-Computer Interaction, vol. 25, 2009, pp. 107-133.

[2] J. Wang and B. J. Calder, "Media engagement and advertising: Transportation, matching, transference and intrusion," Journal of Consumer Psychology, vol. 19, 2009, pp. 546-555.

[3] P. A. Nussbaum, A. Herrera, R. Joshi, and R. Hargraves, "Analysis of Viewer EEG Data to Determine Categorization of Short Video Clip," Procedia Computer Science, vol. 12, 2012, pp. 158-163.

[4] G. Vecchito, L. Astolfi, F. V. Fallani, F. Cincotti, D. Mattia, and et al., "Changes in brain activity during the observation of TV commercials by using EEG, GSR and HR measurements," Brain Topogr, vol. 23, 2010, pp. 165-179.

[5] T. A. Dennis and B. Solomon, "Frontal EEG and emotion regulation: Electrocortical activity in response to emotional film clips is associated with reduced mood induction and attention interference effects," Biological Psychology, vol. 85, 2010, pp. 456-464.

[6] M. Gola, M. Magnuski, I. Szumska, and A. Wrobel, "EEG beta band activity is related to attention and attentional deficits in the visual performance of elderly subjects," International Journal of Psychophysiology, vol. 89, 2013, 334-341.

[7] N. Lee, A. J. Broderick, and L. Chamberlain, "What is 'neuromarketing'? A discussion and agenda for future research," Int J Psychophysiol, vol. 63, 2007, pp. 199-204.

[8] C. Solnais, J. Andreu-Perez, J. Sanchez-Fernandez, and J. Andreu-Abela, "The contribution of neuroscience to consumer research: A conceptual framework and empirical review," Journal of Economic Psychology, vol. 36, 2013, pp. 68-81.

[9] M. J. R. Butler, "Neuromarketing and the perception of knowledge," Journal of Consumer Behaviour, vol. 7, 2008, pp. 415-419.

[10] R. N. Khushaba, C. Wise, S. Kodagoda, J. Louviere, B. E. Kahn, and C. Townsend, "Consumer neuroscience: Assessing the brain response to marketing stimuli using electroencephalogram (EEG) and eye tracking," Expert Systems with Applications, vol. 40, 2013, pp. 3803-3812.

[11] G. Vecchiato, F. V. Fallani, L. Astolfi, J. Toppi, F. Cincotti, and et al., "The issue of multiple univariate comparisons in the context of neuroelectric brain mapping: An application in a neuromarketing experiment," Journal of Neuroscience Methods, vol. 191, 2010, pp. 283-289.

[12] R. G. Munro, "Data Evaluation Theory and Practice for Materials Properties," NIST Recommended Practice Guide, Special Publication 960-11, 2003, pp. 37-43.

[13] A. Urbano, C. Babiloni, F. Carducci, L. Fattorini, P. Onorati, and F. Babiloni, "Evaluation of measurement data - Guide to the expression of uncertainty in measurement," BIPM, JCGM 100:2008, 2010, pp. 4-27.

[14] J. A. Russel, "A Circumflex Model of Affect," J. Pers. Soc. Psychol., vol. 39, 1980, pp. 1161-1178.

[15] G. Betta, C. Liguori, and A.Pietrosanto, "Propagation of uncertainty in a discrete Fourier transform algorithm," Measurement, vol. 27, 2000, pp. 231-239.

 ISBN: 978-1-61208-337-7

A Platform for Secure and Trustworthy Service Composition

Michela D'Errico, Francesco Malmignati
Selex ES S.p.A.
Rome, Italy
{michela.derrico, francesco.malmignati}@guests.selex-es.com

Giovanni Fausto Andreotti
Italtel S.p.A.
Milan, Italy
fausto.andreotti@italtel.com

***Abstract* — The Future Internet is moving from today's static services to an environment in which service consumers will transparently mix and match service components depending on service availability, quality, price and security attributes. This fact poses some challenges in terms of security and trustworthiness that should be guaranteed to the final users. In this paper, we present a platform for secure service design and composition based on the Activiti open-source workflow engine and Business Process Model and Notation (BPMN) extensions for expressing security needs over service specifications. The platform, developed in the realm of the Aniketos FP7 funded project, offers the capability to service designers and service providers to establish and maintain trustworthiness and secure behavior in today's constantly changing service environments. In order to demonstrate the validity of this approach, the use of the platform is shown in a real application scenario in which a security requirement on trustworthiness specified by design needs to be monitored and guaranteed during service execution.**

Keywords-service composition; service design; service deployment; security requirements; trustworthiness.

I. INTRODUCTION

A web service composition is needed when a desired functionality cannot be provided by a single web service. A service designer who wants to create a composite service has to specify the way the atomic web services have to be composed in order to fulfill his objectives. To this aim, service compositions can be modelled as business processes, namely, a set of activities that interoperate to perform a task. In this process-oriented approach, service composition is described using workflow languages and technologies.

In this paper, we present part of the work carried out in the Aniketos funded project [1] that aims to establish and maintain trustworthiness and secure behavior in an Internet service environment. As a result, a platform has been developed in order to support the service designer in performing all the steps needed to create and to manage trustworthy and secure service compositions.

Currently, many Web Service (WS-*) specifications address security concerns, but most of the focus is on secure message exchange [2], and current orchestration and choreography lack support for the specification and enforcement of the security process level requirements. These higher order interactions must become a part of composition.

The Aniketos platform overcomes this by allowing a service designer to specify security needs during the modeling of the composite service and by supporting the service discovery and composition/adaptation based on security properties and not just on the functional descriptors.

The notion of a trustworthy service at the most basic level can be taken as a service satisfying some minimum security requirements, most notably attestation and authorization of service endpoints, and the use of secure communication channels [3], but also involves a judgment about how likely that service is to perform as claimed. At present, a disconnection exists between the diverse mechanisms for managing trust. There is also a lack of solutions for availability and security aspects of dynamic binding (e.g. at runtime) of services. Although trustworthiness aspects can be defined for services, it's a major challenge to define trustworthiness aspects for composite services. Some approaches take into account various factors such as reputations and qualities of the services [4] and confidentiality and integrity [5].

Aniketos offers a way of expressing different aspects of trustworthiness and provide design time and runtime modules for evaluating and monitoring the trust level between service providers/components.

The present paper is organized as follows: Section II is dedicated to the background concepts that have been adopted and further developed to realize the platform. In Section III, an overview of the components of the Aniketos platform and the main features supported by a set of software packages is described. The application of the Aniketos design time tool-chain and service runtime management is reported in Section IV. Section V shows how the Aniketos platform can be used to implement a real case study. Finally, Section VI deals with related works and concludes the paper.

II. COMPOSITE SERVICE MODELING

In this section, components and concepts adopted for the development of the platform are presented.

A. BPMN language

The platform for web service composition developed in Aniketos uses Business Process Model and Notation (BPMN) [6], a de facto standard for the process modeling, together with the Activiti engine [7], a process server able to execute BPMN business processes. The Aniketos platform adds to the service composition the support for security and trustworthiness properties management by extending the BPMN standard language with proper security annotations.

BPMN, being a flow chart based notation, facilitates the modelling of the workflow for the service composition. The resulting business process diagram is easily understandable by technical as well as business users.

 ISBN: 978-1-61208-337-7

The Aniketos platform provides the service designer a set of features for the realization of runnable secure and trustworthy composite services. The whole design time specification process, along with runtime tools available for the management of the service execution, are detailed in the remainder of the paper.

B. BPMN elements

The BPMN specification contains a large set of object types, over 100, but a very small set of the constructs can be used to model a service composition with the Aniketos platform.

Core BPMN elements are grouped into four categories: Flow objects, Connection objects, Swim lanes and Artifacts. Throughout the paper only the elements belonging to the first two categories will be used, specifically Event, Task and Gateway as Flow objects, Sequence Flow as Connection objects. In the following, a brief description of these elements is given:

- *Start event*: represents the event that triggers the start of the service execution.
- *End event*: represents the end of the process execution.
- *Task*: represents part of the work to be done in the process that cannot be further decomposed.
- *Gateway*: is used to model forking and merging of paths.
- *Sequence* Flow: is used to connect the Flow objects and to specify in which order the tasks must be executed.

III. ANIKETOS PLATFORM

An overview of the Aniketos platform is depicted in Figure 1.

Design time support is available for service designers that use the platform in order to build secure and trustworthy service compositions. The capabilities offered in terms of analysis and composition of services are supported by the underlying components of the platform, although in this paper we will focus on the use of the front-end tools for the specification of secure and trustworthy service compositions.

The runtime support is dedicated to the monitoring of service properties during execution and adaptation in case of violation of security specifications.

The Aniketos platform [8] is composed by a number of components that have been grouped into four packages:

- Socio-Technical Security Requirements
- Secure Service Specification, Discovery & Deployment
- Secure Service Validation & Verification
- Security Monitoring & Notification.

Software packages can be potentially targeted to a variety of application domains that need security and trustworthiness, thus, aiming to effective exploitation of Aniketos' results.

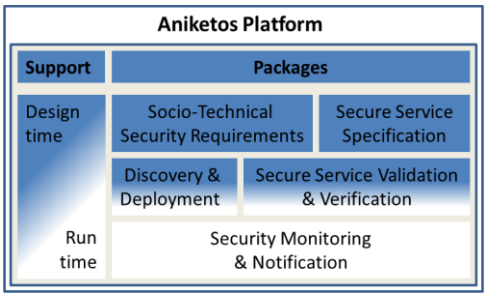


Figure 1: Aniketos platform overview.

The packaging takes into account the components functionalities, their licensing scheme and their role in the composite service process lifecycle.

The originality of this approach resides not only in the complete set of design time and runtime capabilities offered in an unique framework, but is also due to the possibility for the users to selectively adopt the features provided by the Aniketos packages according to their needs, thus, reducing the cost of the final realization.

A. Socio-Technical Security Requirements

This package offers a graphical tool [9] to model Socio-Technical Systems (STSs) [10][11] that are complex systems in which social actors interact with one another and with technical components to fulfill their goals. In such systems, many security issues arise from the interaction between actors and from the manipulation of the exchanged information.

A threat repository included in this package allows the designers to look for potential threats to be taken into account in the model and to acquire useful information on the threats and possible countermeasures in order to mitigate the associated risk.

In the realm of Aniketos, the package can be used to model high level security and trustworthiness requirements of a holistic application and/or parts of it. Of particular interest is the possibility to model the security requirements for a composite service process, which has to be developed from scratch or already exists and needs to conform to specific requirements.

B. Secure Service Specification, Discovery & Deployment

This package is used to model the composite service process with BPMN and to specify the security and trustworthiness requirements that the services taking part in the composition must fulfill. The BPMN is thus enriched with the consumer policies that represent the low level representation of the requirements expressed by the service designer through the Socio-Technical Security Requirements package. The package offers the possibility to publish Aniketos compliant services to the Aniketos *Marketplace*, an enriched service registry that supports discovery of atomic services and provision of security descriptors, also called agreement templates.

Finally, this package enables the deployment of the created service compositions to an application server for runtime execution.

 ISBN: 978-1-61208-337-7

Currently, a set of security properties are supported, namely trustworthiness, separation and binding of duty, confidentiality, non-repudiation, integrity and need-to-know. In the remainder of the paper, the trustworthiness management will be described in detail.

C. Secure Service Validation & Verification

This package offers verification and validation checks during design, announcement and execution of secure services.

This mechanism is built upon the matching between the consumer policies, representing the desired properties, and the services agreement templates, representing the properties that can be provided by the services.

The service validation process can be invoked when a composite service has been designed and the service developer needs to ensure that the security features of the involved services comply with the service specification.

The same check can be performed at runtime to validate that the offered security level of the composite service keeps complying with the consumer's security policy.

Furthermore, this package is used to perform a thorough security check on the properties declared by the components of a composite service.

D. Security Monitoring & Notification

This package enables the monitoring of execution of composite services and the generation of alerts when any malfunction in the proper service operation is identified.

Such malfunctions can refer to the violation of a service contract and/or the change in the trustworthiness level and/or detection of threat affecting the offered composite service.

The package enables subscriptions to service monitoring modules in charge of capturing and analyzing specific type of events produced by the service execution environment and of generating alerts and notifications for potential breaches at the service layer.

IV. SECURE SERVICE COMPOSITION FRAMEWORK

The *Service Composition Framework* (SCF) is the design time tool for secure and trustworthy service composition and is based on Activiti designer [7]. The SCF includes the functionalities of the *Secure Service Specification, Discovery & Deployment* package and the *Secure Service Validation & Verification* package, in order to offer an integrated and complete environment allowing the service designer to perform all the steps needed to create a composite web service. The design process starts with building the BPMN model of the composite service as a business process and ends with the deployment of the created composition plan as a web service. The deployment entails the announcement of the service to a Marketplace so that it can be made available and discovered by service providers.

A. Business process modeling

A composite service is a complex service made up of atomic services that can be connected in different ways, thus, providing different results based on how the services are combined.

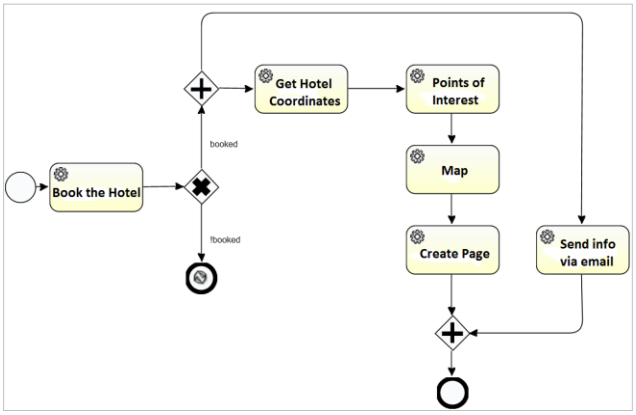


Figure 2: Service composition business process.

A business process starts with a *Start Event* element and ends with an *End Event* element. A web service composition can be modeled by using a *Start Event*, a set of service tasks connected with gateways and connection elements and an *End Event*, as shown in Figure 2. The tasks used for the service composition are service tasks that identify a piece of the process that is executed by invoking a web service.

The way the tasks are connected specifies how the final output of the composite service is built by using the output provided by the service tasks. The BPMN diagram is the graphical representation of the service composition. An example is shown in Figure 2.

B. Binding and discovery of services

In order to produce a runnable composition plan, each service task has to be bound to a web service. The binding has to be chosen to satisfy the functional requirements assigned to the service task. To this aim, for each service task, the *Type* parameter must be set to allow the discovery of well-suited web services. The value for the *Type* parameter is chosen from a service taxonomy agreed and shared by service designers and developers. The taxonomy is provided along with a vocabulary explaining what each service type means and entails in terms of the provided functionality.

In order to help the service designer to set the service *Type*, the SCF provides in a pop-up a Type Cloud showing the set of the service types that are available in the *Marketplace*.

Once specified the service *Type* for each service task the SCF is ready to execute the service discovery, which will return the set of operations provided by web services belonging to the service *Type* category specified. The SCF shows, for each operation, the input required and the output provided. For each service task, the service designer has to select one of the available web service operations and has to specify a valid input in terms of process variable or directly defining it with a plain value.

Moreover, for each service task, the service designer has to create the variable that will contain the result of the task, that is the output provided by the selected web service operation.

 ISBN: 978-1-61208-337-7

The set of process variables available for each service task contains the output variables of all the service tasks executed before in the process and the variables provided by the Start Event. The set of input added to the Start Event can be seen as global variables accessible by any task in the process.

C. Security and trustworthiness requirements

The binding of service tasks with web services has the aim to fulfill the functional requirements. The security and trustworthiness requirements can be specified once the binding has been completed. As for the properties specification at BPMN level, we will focus on confidentiality, integrity and trustworthiness.

The service designer can add confidentiality property if data to be transmitted between service tasks have to be kept confidential and thus the use of an encryption algorithm is required. The service designer can also specify which data (input, output or both) are required to be enciphered.

The integrity property can be added to specify that the service designer needs that a "sender" service has to apply a mechanism enabling the "receiver" service to detect whether data has been corrupted or modified by an attacker. In this way, the designer can specify the mechanism and the algorithm to be used by the "sender" service.

The trustworthiness property is specified by setting a threshold value that is computed [12] by taking into account a set of parameters such as service provider's reputation and Quality of Service (QoS) [13].

The specified properties represent the consumer policy, namely the desired properties that have to be compared with the provided security properties to establish whether the web services selected during the discovery phase satisfy the security and trustworthiness requirements. The offered properties are encoded in *ConSpec* language [14] and written in a file called agreement template, associated to each web service in the Aniketos *Marketplace*.

To enable the comparison, the consumer policies have to be encoded in *ConSpec* language as well by using an editor available in the SCF. In this operation, the SCF helps the service designer with a *ConSpec* editor that will use the properties specified in the previous step to automatically fill out the consumer properties files.

D. Creation and validation of composition plans

After binding service tasks with web services and specifying security requirements, the SCF is used to create a set of composition plans. The operation selected for service task S_k during the discovery phase can be offered by $N_{ws}(k)$ different web services, this means that N_{cp} composition plans can be created satisfying the functional requirements. The number N_{cp} of possible composition plans is given by the following formula:

$$N_{cp} = \prod_{1}^{N} N_{ws}(k) \quad (1)$$

where N is the number of service tasks in the composition and $N_{ws}(k)$ is the number of web services providing the web operation selected for the binding of the k^{th} service task.

At this point, the service designer can use the SCF to start the security and trustworthiness verification of the composition plans. The result of this verification will be the set of composition plans having web services whose offered properties match the consumer policies.

E. Rules definition and service deployment

Then, the service designer has to select one of the composition plans and has to specify the rules for the management of events that can occur during runtime execution. A set of rules can be defined to handle specific events, such as a threat detection or a violation of the security and trustworthiness properties specified during design. For each rule, the service designer can specify the constraints for the event to fire the rule and the action to be performed once the rule is fired. For example, an action we introduce here is the recomposition, a mechanism that replaces the web service offering the operation selected during the binding. The result of this action will be a different runnable composition plan satisfying the same security and trustworthiness properties as the substituted web service.

Having specified the rules, the designer can complete the design time process by deploying the composite service. The deployment entails the exposure of the composition as a web service and its announcement in the Aniketos *Marketplace*, thus, allowing other service designers to use it as part of other service compositions.

F. Runtime support

The platform includes a Service Runtime Environment (SRE) that is in charge of executing the services and enforcing the rules specified by the service designer. To this aim the SRE must interact with the modules that monitor the services security and trustworthiness. Specifically, based on the events specified in the rules, the SRE subscribes to the modules that are able to detect and notify about those specific events. This mechanism allows the SRE to trigger the action specified in the rule when the related event is received.

Support for runtime management is provided by the *Security Monitoring & Notification* package.

V. APPLICATION TO A REAL CASE STUDY

In this section, we describe the application of the Aniketos design time and runtime support to a real example taken from an industrial case study [15]. The aim is to create a *HotelBookingService* that takes as input the user and hotel data and returns the reservation detail. The service will then be used to build a web application for hotel reservations. The process is split into the following main phases:

a) Design time: the designer creates a composite service with a trustworthiness security requirement. Before deploying the service the designer specifies that, if at runtime the requirement is not fulfilled, the service has to be

 ISBN: 978-1-61208-337-7

recomposed. Then, the service is deployed and integrated into a web application that is available to the user.

b) Runtime: the system performs a service adaptation (in this specific case a recomposition) of the service in case of security violations detected during service execution, in a manner that is totally transparent to the user.

B. Scenario definition

The service designer aims to create a hotel reservation service that takes in input user's preferences and data that are necessary for booking the hotel and returns a web page with Points of Interest (POIs) related to the hotel location plus a confirmation mail. The scenario is depicted in Figure 3.

The composite service includes an atomic service that provides a map showing a set of locations. The designer selects the service offered by Service Provider A (SP-A) and the resulting composite service is deployed and integrated into a web application.

An incident happens to the servers of the SP-A: this affects the reputation of the service provider and thus the trustworthiness associated to it. The final result is that the trustworthiness becomes lower than the threshold value set by the designer at design time implying that the service properties do not match anymore the consumer policies and thus a recomposition is needed. The result will be the substitution of the atomic service provided by SP-A with a similar service provided by another Service Provider (SP-B) matching the trustworthiness requirement.

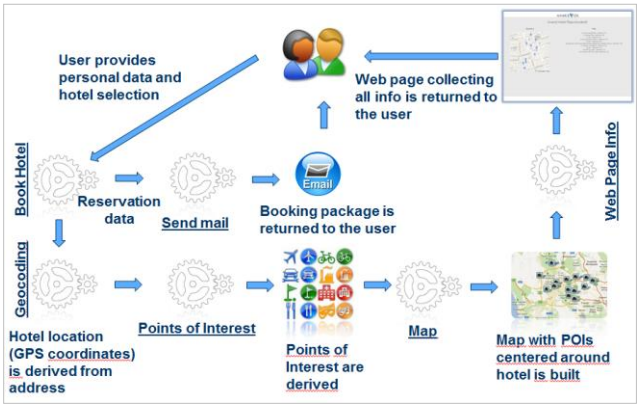


Figure 3: Application scenario.

C. Creation of Composite Service

This section aims to describe the use of the Aniketos front-end tools following the process described in Section IV for the design of a secure and trustworthy composite service in the application scenario.

The process starts in Figure 4 with drawing the business process representing the composite service. Then, the following operations are performed in sequence: discovery, configuration and specification of the security and trustworthiness requirements over the services involved in the composition.

The requirements that are needed in this case are: *confidentiality*, *scope of usage* and *trustworthiness*.

- Confidentiality will be needed to transmit data related to user's information in a secure manner.
- Scope of usage (e.g., need-to-know) will guarantee that user's data will be used only in the scope of the service and not for any other purposes.
- Trustworthiness indicates that a (minimum) level of trustworthiness value for elementary services (*BookingSrv and MapSrv*) that belong to the composition is required.

Since trustworthiness will be the only requirement to be monitored in this scenario, it's worth to describe that in Aniketos the trustworthiness of the composite service is evaluated by using the weakest link principle. A specific module of the Aniketos platform (Trustworthiness module) evaluates the trustworthiness value for each service taking part in the composition [16]: the lowest value is returned as the trustworthiness value of the composite service. In Aniketos, the trustworthiness value is a combination of cognitive and non-cognitive measure of trust [17][18].

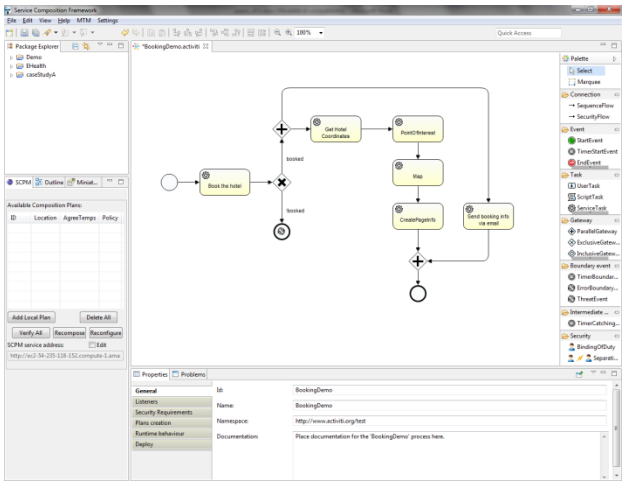


Figure 4: Service Composition Framework.

When the modeling process is complete, the composition plans are generated and validated in terms of trustworthiness.

Finally, the service designer makes a selection and deploys a specific composition plan in order to make it available to other service providers through the Marketplace.

D. Service-based application

This section describes how the deployed service will be used in the final application.

A web service is provided with a Web Service Description Language (WSDL) [19] that exposes the operations offered by the service and gives information on how the operations can be invoked by a client. This means that the service result cannot be taken as it is but has to be integrated, for instance, into an application. For the hotel reservation application, the service client is integrated in a web page. The input required by the service is asked through a form that is submitted by the user. Thus, the composite service execution is triggered and the result is presented in a new web page returned by the composite service.

 ISBN: 978-1-61208-337-7

E. Security management at runtime

This section shows the runtime execution of the composite service including the monitoring for detection of contract violations. In this scenario, when the level of trustworthiness is no longer guaranteed by an atomic service belonging to the composition, an event detecting this contract violation is sent to the runtime environment.

According to the rule set at design time, a recomposition is triggered and leads to the substitution of the atomic service with another one matching the trustworthiness requirement. Figure 5 shows the application rendering when the atomic service that provides the map is replaced.

Figure 5: Application rendering.

This simple scenario demonstrates the effectiveness of the Aniketos approach that is to establish and maintain the security and trustworthiness properties of the service during service execution, in a totally transparent manner to the end-user.

VI. CONCLUSION AND FUTURE WORK

A platform for designing and ensuring secure and trustworthy service compositions has been presented. This approach, developed in the realm of the Aniketos project, covers all the phases in the service development chain, ranging from modeling and specification of security needs to the actual operation of the delivered services.

The Aniketos platform uses design time descriptions to establish trust and verify safe and secure service behaviour among several different service providers. Runtime monitoring and automatic adaptation of services are needed due to an evolving environment of threats and operating conditions. Down-time is costly; a composed service must be able to operate even during an attack (with possible limitations or change of behaviour), taking risks and adaptation costs into account.

Further improvements are needed to make discovery and security verification more mature, and to express what the compositions should do in case of attacks. Although some existing security modelling and verification techniques allow the service composer to specify security properties, the number of services satisfying these requirements may be large. It will be therefore important to enhance the Aniketos platform and offer a scalable approach to service compositions based on security properties.

Aniketos already delivers a functional service runtime environment that supports reception of warnings from a notification service and dynamically adapts the composition in a risk-reducing manner. However, further work is needed to enlarge the set of monitored security properties.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no. 257930 [1].

REFERENCES

[1] Aniketos: Ensuring Trusworthiness and Security in Service Composition, http://www.aniketos.eu [retrieved: April, 2014].

[2] A. Charfi et al., "Reliable, Secure, and Transacted Web Service Compositions with AO4BPEL," in Proc. of the European Conference on Web Services, 2006, pp. 23-34.

[3] Z. Jianwu et al., "On achieving trustworthy SOA-based Web Services," SAM'06, Las Vegas, NV, USA, 2006, pp. 341-347.

[4] H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness monitoring and prediction of composite services", ISCC, 2012, pp.580–587.

[5] B. Zhou et al, "Secure service composition adaptation based on simulated annealing", ACSAC, 2012, pp. 49–55.

[6] OMG, Business Process Model and Notation (BPMN), 2011, http://www.omg.org/spec/BPMN/2.0.

[7] Activiti BPM Platform, http://www.activiti.org [retrieved: April, 2014].

[8] Aniketos Deliverable 5.3 "Final Aniketos Platform integration", March 2014.

[9] Socio-Technical Security modeling language and tool, http://fmsweng.disi.unitn.it/sts [retrieved: April, 2014].

[10] F. E. Emery and E. L. Trist, "Socio-Technical Systems", Management Sciences, Models and Techniques, editors Churchman, C. W. Pergamon, London, 1960.

[11] F. Dalpiaz et al, "Security requirements engineering via commitments", IEEE STAST, 2011, pp. 1-8.

[12] H. Elshaafi, and D. Botvich, "Aggregation of trustworthiness properties of BPMN-based composite services.", IEEE CAMAD, 2012, pp. 383-387.

[13] E. Maximilien and M. Singh, "Agent-based trust model involving multiple qualities", Proc. 4th Int. Conf. on AAMAS, 2005.

[14] I. Aktug and K. Naliuka, "ConSpec - A Formal Language for Policy Specification", Electr. Notes Theor. Comput. Sci., (197) 1, 2008, pp. 45-58.

[15] Aniketos Deliverable 6.1 "Initial analysis of the industrial case studies", July 2011.

[16] H. Elshaafi, J. McGibney, and D. Botvich, "Trustworthiness monitoring and prediction of composite services", IEEE ISCC, 2012, pp. 580-587.

[17] Aniketos Deliverable 2.1 "Models and methodologies for embedding and monitoring trust in services", July 2011.

[18] Aniketos Deliverable 2.4 "Models and methologies for integrated security and trust paradigm for service contracts", June 2013.

[19] E.Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", World Wide Web Consortium (W3C), 2001.

 ISBN: 978-1-61208-337-7