IJSC Editorial Board

Editors-in-Chief
Andrzej Goscinski, Deakin University, Australia
Jia Zhang, Carnegie Mellon University – Silicon Valley, USA

Associate Editor-in-Chief
Zibin Zheng, The Chinese University of Hong Kong

Editorial Board
Boualem Benatallah, University of New South Wales, Australia
Rajdeep Bhowmik, Cisco Systems Inc., USA
Irena Bojanova, University of Maryland University College, USA
Hong Cai, ZTE, USA
Rong Chang, IBM T.J. Watson Research Center, USA
Shiping Chen, CSIRO, Australia
Chi-Hung Chi, CSIRO, Australia
Ying Chen, Wisedu Information Technology, China
Ernesto Damiani, University of Milan, Italy
Schahram Dustdar, Vienna University of Technology, Austria
Ephraim Feig, Independent Consultant, USA
Aditya Ghose, University of Wollongong, Australia
Claude Godart, Univ. Nancy, Lorraine Univ., France
Nils Gruschka, University of Applied Science Kiel, Germany
Mohand-Said Hacid, Université Claude Bernard Lyon 1, France
Jun Han, Swinburne University of Technology, Australia
Akhil Kumar, Pennsylvania State University, USA
Patrick Hung, University of Ontario, Canada
San-Yih Hwang, National Sun Yat-sen University, Taiwan
Geetika Lakshmanan, IBM T.J. Watson Research Center, USA
Donghui Lin, Kyoto University, Japan
Shiyong Lu, Wayne State University, USA
Michael Lyu, The Chinese University of Hong Kong, Hong Kong
Stephan Marganiec, University of Leicester, UK
Louise Moser, University of California Santa Barbara, USA
Lorenzo Mossucca, Istituto Superiore Mario Boella(ISMB), Italy
Sushil Prasad, Georgia State University, USA
Lakshmish Ramaswamy, University Of Georgia, USA
Krishna Ratakonda, IBM Watson Research Center, USA
Anna Ruokonen, Penn State University, USA
George Spanoudakis, City University London, UK
Vijay Varadharajan, Macquarie University, Australia
Wei Tan, IBM T.J. Watson Research Center, USA
Mingdong Tang, Hunan University of Science and Technology, China
Hongbing Wang, Southeast University, China
Shangguang Wang, Beijing University of Posts and Telecommunications, China
Yan Wang, Macquarie University, Australia
Jian Wu, Zhejiang University, China
Xiaofei Xu, Harbin Institute of Technology, China
Yuhong Yan, Concordia University, Canada
Bing Bing Zhou, The University of Sydney, Australia

http://hipore.com/ijsc
**Call for Articles**

**International Journal of Services Computing**  
ISSN: 2330-4472  
http://www.hipore.com/ijsc/

**Mission**

The International Journal of Services Computing (IJSC) aims to be a reputable resource providing leading technologies, development, ideas, and trends to an international readership of researchers and engineers in the field of Services Computing. To ensure quality, IJSC only considers as candidates extended versions of papers published at reputable international conferences such as IEEE ICWS, and at least two IJSC Editorial Board members will review the extended versions.

From the technology foundation perspective, Services Computing covers the science and technology needed for bridging the gap between Business Services and IT Services, theory and development and deployment. All topics regarding Web-based services lifecycle study and management align with the theme of IJSC. Specially, we focus on: 1) Web-based services, featuring Web services modeling, development, publishing, discovery, composition, testing, adaptation, and delivery, and Web services technologies as well as standards; 2) services innovation lifecycle that includes enterprise modeling, business consulting, solution creation, services orchestration, services optimization, services management, services marketing, business process integration and management; 3) cloud services featuring modeling, developing, publishing, monitoring, managing, delivering XaaS (everything as a service) in the context of various types of cloud environments; and 4) mobile services featuring development, publication, discovery, orchestration, invocation, testing, delivery, and certification of mobile applications and services.

**Topics**

The International Journal of Services Computing (IJSC) covers state-of-the-art technologies and best practices of Services Computing, as well as emerging standards and research topics which would define the future of Services Computing. Topics of interest include, but are not limited to, the following:

- Services Engineering
- XaaS (everything as a service)
- Cloud Computing for Internet-based services
- Big Data services
- Internet of Things (IoT) services
- Pervasive and Mobile services
- Social Networks and Services
- Wearable services
- Web 2.0 and Web X.0 in Web services
- Service-Oriented Architecture (SOA)
- RESTful Web Services
- Service modeling and publishing
- Service discovery, composition, and recommendation
- Service operations, management, and governance
- Services validation and testing
- Service privacy, security, and trust
- Service deployment and evolution
- Semantic Web services
- Scientific workflows
- Business Process Integration and management
- Service applications and implementations
- Business intelligence, analytics and economics for Services

http://hipore.com/ijsc
# Table of Contents

## Editor-in-Chief Preface

**EDITOR-IN-CHIEF PREFACE**

iv

Andrzej Goscinski, Deakin University, Australia  
Jia Zhang, Carnegie Mellon University – Silicon Valley, USA

## Research Articles

### Using Finite State Models for Quality Evaluation at Web Service Development Steps

Olga Kondratyeva, TELECOM SudParis, Evry, France; Tomsk State University, Tomsk, Russia  
Natalia Kushik, TELECOM SudParis, Evry, France; Tomsk State University, Tomsk, Russia  
Ana Cavalli, TELECOM SudParis, Evry, France  
Nina Yevtushenko, Tomsk State University, Tomsk, Russia

### An Event-Based Approach for Declarative, Integrated and Self-Healing Web Services Composition

Ehtesham Zahoor, National University of Computer and Emerging Sciences, Pakistan  
Kashif Munir, National University of Computer and Emerging Sciences, Pakistan  
Olivier Perrin, Université de Lorraine, LORIA, France  
Claude Godart, Université de Lorraine, LORIA, France

### A Digital Security Certificate Framework for Services

Samuel Paul Kaluvuri, SAP Labs France  
Hristo Koshutanski, University of Malaga, Spain  
Francesco Di Cerbo, SAP Labs France  
Renato Menicocci, Fondazione Ugo Bordoni, Italy  
Antonio Maña, University of Malaga, Spain

### Formal Methods for the Specification and Testing of Data-Centric Web Services: A Case Study

Iman Saleh, University of Miami, USA  
Gregory Kulczycki, Virginia Tech, USA  
M. Brian Blake, University of Miami, USA  
Yi Wei, Microsoft, USA

### A Location-Aware Service Selection Model

Ying Li, Zhejiang University, China  
Zhiling Luo, Zhejiang University, China  
Jianwei Yin, Zhejiang University, China

### Call for Papers: IEEE CLOUD/ICWS/SCC/MS/BigData/SERVICES 2014  
Call for Articles: International Journal of Cloud Computing (IJCC)  
Call for Articles: International Journal of Big Data (IJBD)
Editor-in-Chief Preface:
Services Discovery and Management
Andrzej Goscinski, Deakin University, Australia
Jia Zhang, Carnegie Mellon University – Silicon Valley, USA

Welcome to the inaugural issue of International Journal of Services Computing (IJSC). From the technology foundation perspective, Services Computing covers the science and technology needed for bridging the gap between Business Services and IT Services, theory and development and deployment. All topics regarding Web-based services lifecycle study and management align with the theme of IJSC. Specially, we focus on:

1. Web-based services featuring Web services modeling, development, publishing, discovery, composition, testing, adaptation, and delivery; and Web services technologies as well as standards;
2. services innovation lifecycle that includes enterprise modeling, business consulting, solution creation, services orchestration, optimization, management, and marketing; and business process integration and management;
3. cloud services featuring modeling, developing, publishing, monitoring, managing, and delivering XaaS (everything as a service) in the context of various types of cloud environments; and
4. mobile services featuring development, publication, discovery, orchestration, invocation, testing, delivery; and certification of mobile applications and services.

IJSC is designed to be an important platform for disseminating high quality research on above topics in a timely manner and provide an ongoing platform for continuous discussion on research published in this journal. To ensure quality, IJSC only considers expanded version of papers presented at high quality conferences, key survey articles that summarizes the research done so far and identifies important research issues, and some visionary articles. At least two IJSC Editorial Board members will review the extended versions. Once again, we will make every effort to publish articles in a timely manner.

This inaugural issue collects the extended versions of five IEEE ICWS articles in the general area of services discovery and management.

The first article titled, “Using Finite State Models for Quality Evaluation at Web Service Development Steps” by Kondratyeva, Kushik, Cavalli, and Yevtushenko, studies how to leverage finite state models to enhance the accuracy of Quality of Service (QoS) and Quality of Experience (QoE) of Web services. A use study is reported.

The second article titled, “An Event-Based Approach for Declarative, Integrated and Self-Healing Web Services Composition” by Zahoor, Munir, Perrin, and Godart, addresses how to use formalism to facilitate service composition. They present an event-oriented framework supporting service composition design, verification and monitoring.

The third article titled, “A Digital Security Certificate Framework for Services” by Kaluvuri, Koshutanski, Di Cerbo, Menicocci, and Maña, tackles the issue of assuring service security. The authors propose a language for machine processable digital security certificates (DSC), and introduce a concept of security profile associated with a tool to generate uniform DSCs.

The fourth article titled, “Formal Methods for the Specification and Testing of Data-Centric Web Services: A Case Study” by Saleh, Kulczycki, Blake, and Wei, tackles the issue of Service Level Agreement (SLA). The authors present a case study formally specifying and verifying data-centric Web services. Automatic test case generation is discussed as well.

The fifth article titled, “A Location-Aware Service Selection Model” by Li, Luo, and Yin, advocates to
consider service location as an important criterion in service selection. The evaluation function of service selection is modeled as a Simple Additive Weighting (SAW) problem with optimization algorithms established. A case study is presented to illustrate the need and use of the proposed work.

We would like to thank the authors for their effort in delivering these five quality articles. We would also like to thank the reviewers, as well as the Program Committee of IEEE ICWS for their help with the review process. Finally, we are grateful for the effort Liang-Jie Zhang made in giving birth to this inaugural issue of International Journal of Services Computing (IJSC).

About the Editors-in-Chief

Dr. Andrzej Goscinski is a full Professor in the School of Information Technology, Deakin University, Australia, where he directs research programs in clouds and cloud computing, parallel processing, virtualization, security, autonomic and service computing, and in general, distributed systems and applications. From January 1993 to December 2001, Dr. Goscinski completed tenure as the Head of School, and from 2004 he has led his research group to successfully concentrate their research on autonomic grids based on SOA, the abstraction of software and resources as a service, and cloud computing. A major achievement in the area of autonomic grids based on SOA was the development of the concept of a broker that led to its use in clouds. Furthermore, a major achievement in the area of the abstraction of software and resources as a service and cloud computing was the development of the Resource Via Web Services (RVWS) framework that contains service’s dynamic state and characteristics, and service publishing, selection and discovery; the contribution to level of cloud abstraction in the form of CaaS (Cluster as a Service); comparative study of High Performance Computing clouds, and the development of H2D hybrid cloud. Currently, he concentrates his research on exposing HPC applications as services, publishing them to a broker, and executing them in a SaaS cloud by non-computing specialists. The results of this research have been published in high quality journals and conference proceedings. Dr. Goscinski serves as Associate Editor of IEEE Transactions on Service Computing; Associate Editor of Inderscience’s International Journal on Cloud Computing; member of the Editorial Board of Springer's Future Generation Computer Systems; and General, Program Chair, and Honorary Chair of IEEE Services and Cloud Conferences, and Distributed and Parallel Systems and Applications.

Dr. Jia Zhang is an Associate Professor at Carnegie Mellon University - Silicon Valley. Her recent research interests center on services computing, with a focus on scientific workflows, Internet of Things, net-centric collaboration, and cloud-based big data analytics. She has co-authored one textbook titled “Services Computing” and has published over 120 refereed journal papers, book chapters, and conference papers. She is now an Associate Editor of IEEE Transactions on Services Computing (TSC) and of International Journal of Web Services Research (IJWSR). Zhang is Program Vice Chair of IEEE International Conference on Web Services (ICWS). She earned her Ph.D. in computer science from the University of Illinois at Chicago.
Using finite state models for quality evaluation at web service development steps

Olga Kondratyeva¹², Natalia Kushik¹², Ana Cavalli¹, Nina Yevtushenko²
¹TELECOM SudParis, Evry, France; ²Tomsk State University, Tomsk, Russia
olga.kondratyeva@telecom-sudparis.eu, ngkushik@gmail.com, ana.cavalli@telecom-sudparis.eu, ninayevtushenko@yahoo.com

Abstract
This paper presents a short study on evaluating the web service quality based on finite state models using the well-known metrics of Quality of Service (QoS) and Quality of Experience (QoE). These metrics represent objective and subjective assessments. The development of any web service involves several steps and the final values of the service QoS and QoE can be essentially improved/deteriorated based on the quality reached at each development step. In this paper, five service development steps are considered: those are requirements specification, provisioning, composition, implementation, and usage and management. At each step different finite state models for refining the service quality estimation are utilized. A running example of a vacation planner service illustrates the application of finite state models to improve the accuracy of the QoS and QoE evaluation.

Keywords: web service; quality of service (QoS); quality of experience (QoE); finite state models; attribute/parameter; composite service quality.

1. INTRODUCTION

Nowadays web services are used almost everywhere (Al-Masri & Mahmoud, 2008). Moreover, the number of services increases very fast in order to allow users an easy manipulation of various online applications. Different services developed for the same purpose can be found in service repositories (see, for example, Curbera et al., 2002). Nevertheless, the same functional properties do not mean that those web services have the same quality. Thus, in order to efficiently select the best web service among the great number of available services it is necessary to have the adequate evaluation of the service quality.

In most papers, the quality of a given web service is defined as a set or a pattern of attributes/parameters of this service (Khirman & Henriksen, 2002; Al-Masri & Mahmoud, 2008; Hyun-Jong et al., 2008; Morais & Cavalli, 2012). As it is mentioned in (Hyun-Jong et al., 2008), the major attributes to define the Quality of Service (QoS) are the time delay, the package loss percentage, the service access facility (the availability), the reliability, etc. All these parameters are rather objective and thus, can be evaluated automatically when a set of possible values is specified in advance for each parameter. Nevertheless, there are other attributes that also affect the user satisfaction with a web service, such as service design, how easy the service is to use, etc. Those parameters are subjective and their values significantly depend on human preferences. Those informal service requirements together with QoS parameters are usually evaluated as the Quality of Experience (QoE) for the web service. In other words, when measuring or predicting the user satisfaction or User Experience (Winckler, Bach, & Bernhaupt, 2013) with respect to a web service both QoS and QoE should be taken into account.

In this paper, we focus on formal models for the web service quality evaluation, since formal models are now widely used to evaluate the quality of different software products. Web services can often be described by sets of permissible sequences of actions and finite state models have been proven to be well adapted to verify functional properties of web applications. In this study, finite state models are shown to be very helpful when evaluating the quality of web services.

As any software product, a web service passes different steps while being developed. Thus, web service quality can be eventually improved/deteriorated at each development step. This is the reason why a precise quality evaluation needs to be performed at each service development step. In this paper, we consider the following development steps: service requirements specification, provisioning, composition, implementation, and usage and management.

At the first two steps, the specification and provisioning steps, a set of service requirements as well as a set of available resources needed for the service implementation are determined. The service semantics and external interfaces are given by a formal language description of a service (Booth et al., 2004). One of the basic standard languages is the web service description language (WSDL) and a number of service depositories publish WSDL service files and provide the automatic analysis of such files (see, for example, XMethods depository).
However, in most cases, web services are represented as a composition of other available services. At the service composition step a complex service is designed as a composition of simpler ones. Special languages for the composite service description have been developed. Workflows which support the execution logic of composite complex services can be described using the business process execution language (BPEL). A service implementation significantly depends on a composition derived at the service composition step. Correspondingly, the composition and implementation steps are considered in the same section.

Different techniques can be applied to estimate the QoS at the specification, provisioning, composition and implementation service development steps. In this paper, we focus on using finite state models for this purpose and discuss how different models are used at each service development step. For instance, we discuss that finite automata describing service functionality can be augmented with weights to take into account the service provisioning. We further show that sometimes workflows, which are often used for representing a composite service, are insufficient for the precise evaluation of its quality. At the last step of web service development, the usage and management step, the service becomes available for users and is adapted to user preferences, if needed. Thus, at this step basically the QoE is evaluated. The QoE estimation problem mainly refers to a problem of modeling user perception that is nowadays one of the most well-known problems in the artificial intelligent area. Different finite state models might be of a big help when predicting the QoE value of a given web service. Based on the QoE prediction using the developed models, decisions could be made for which groups of users the service can be attractive and what profit can be obtained. Another issue can be concerned about why the service is not attractive for some groups of users and what should be modified in order to obtain a higher profit of this service usage.

This paper is an extension of (Kondratyeva, et al., 2013a; Kondratyeva, et al., 2013b) where finite state models have been considered for the QoS and QoE evaluation. The main contribution of this paper is the study how the web service quality estimation can be refined at different development steps using finite state models. Each service development step and corresponding quality estimation technique are illustrated by using an example of a composite service that guides the paper.

The rest of the paper is organized as follows. Section 2 contains the preliminaries. Section 3 is devoted to evaluating the QoS of web services using finite state models at the specification and provisioning steps. Section 4 contains a discussion on the QoS estimation during the composition and implementation development steps. Section 5 is devoted to the brief description of finite state models for the QoE evaluation that are used at the usage and management service development step. Section 6 concludes the paper.

2. PRELIMINARIES

A web service can be defined as a composition of web applications where a server (client) in one application can be turned into a client (server) in another one. A similar definition is given, for example, in W3C technical report (Booth et al., 2004), where a web service is defined as “a software system designed to support interoperable machine-to-machine interaction over a network”.

The QoS (Quality of Service) can be defined as a set of attributes (or parameters), such as the response time, availability, reliability, etc., corresponding to a given web service and allowing to compare and rank services with similar functionalities (Khirman & Henriksen, 2002; Al-Masri & Mahmoud, 2008; Hyun-Jong et al., 2008; Morais & Cavalli, 2012). This set of attributes is often mapped into a single value, quality score, using an appropriate computable function and the result of this function can be an integer, a rational, a (fuzzy) logic constant, etc. The QoS parameters are rather objective and thus, can be evaluated automatically when a set of possible values is specified in advance for each parameter.

Despite of the fact that objective quality attributes are important for having a good service, they are insufficient to guarantee that a user is satisfied with the service, since user satisfaction may be essentially affected by some other attributes, such as service design, ease of use, etc. Those parameters together with the satisfaction itself are highly subjective and significantly depend on user preferences, psychological and physiological features of human perception, his/her mood and other hard predictable reasons. The subjective evaluation of the received service is usually represented with the Quality of Experience (QoE). For better quality-aware management of web services, researchers study the automatic QoE estimation and prediction based on the values of QoS attributes (Lalanne, Cavalli, & Maag, 2012).

Running example. In order to illustrate how finite state models can be used at different development steps for the quality evaluation, an example of a vacation planner service is considered throughout the paper. This example with slight modifications is taken from (Escobedo Del Cid, 2011; Gaston & Le Gall, 2012). The service allows a user to purchase flight tickets and to book an accommodation at the destination point. A user submits traveling dates and the planner proposes a number of available options for flight tickets and hotel rooms. If the user and planner agree on the flight ticket and hotel room then the vacation is successfully booked. Otherwise, the vacation reservation has failed.

Service development steps. As mentioned in the introduction, web services are developed in several major steps which can be defined in different ways starting from classical software development life cycle models (ISO/IEC,
Section 3.1 discusses how the quality requirements of the service under development can be selected. Section 3.2 focuses on the relationships between selected requirements as this is important when comparing the quality of two services with the similar functionalities. In Section 3.3, it is briefly discussed how finite state models extracted from the service functional description can be used for the quality evaluation. In Section 3.4, the extracted formal model is augmented with weights according to the available information about service provider and/or the experience of using services with similar functionalities. Such augmentation sometimes allows a more precise quality evaluation.

3.1 Specifying Web Service Requirements

The requirements for the service under construction can be implicitly divided into two groups: functional and non-functional requirements. In this paper, the requirements that affect the service quality are studied. Correspondingly, at the first step it is important to specify which quality parameters are crucial for the service quality and this choice essentially depends on web service features. The choice of quality parameters has a significant impact on the further attractiveness of the constructed service.

Example. For the vacation planner the following functional requirements can be set. 1) Vacation is booked if both, a flight ticket and a hotel room, are reserved. 2) Flight tickets are proposed before booking a hotel room.

The profit of the vacation planner essentially depends on how fast the requests can be processed and on the availability and popularity of the service. Correspondingly, the response time, service availability and service popularity are considered as crucial parameters in this example. In the running example, after marketing analysis it can be stated that 1) each user request has to be processed in at most 30 seconds, and 2) at least 20% of users who are interested in vacation planning should book the vacation using this service.

In the current section, the formalization of quality requirements is discussed. Functional requirements are formalized in the corresponding specification (Section 3.3). The list of crucial parameters is usually defined as the quality vector $QoS = (q_1, q_2, ..., q_n)$. The restrictions on parameter values (if there are any) can be, for instance, stated by a system of equations or inequalities. Equivalence and order relations can be used for specifying the priorities over quality parameters (Xin et al., 2013). For example, the notations below can be used:

- $q_i > q_j$ – parameter $q_i$ is more important than parameter $q_j$;
- $q_i > > q_j$ – parameter $q_i$ is much more important than parameter $q_j$;
- $q_i \approx q_j$ – parameters $q_i$ and $q_j$ have the same importance.
Example. For the vacation planner the quality vector is $QoS = (t, a, r)$, where $t$ denotes the response time, $a$ is the availability and $r$ is the popularity. According to the quality requirements and marketing analysis, those parameters can be formalized in the following way:

\[
\begin{align*}
    t &\leq 30 \\
    r &\geq 0.2 \\
    t &> r \\
    t &> a
\end{align*}
\]

The system (1) establishes the limitations on the values of the response time and popularity, while the system (2) establishes a partial order relation on the set of parameters and states that the response time has the priority with respect to the availability and popularity.

3.2 Comparing the Quality of Two Services

To be able to produce a “better” service than those which already exist, it is necessary to define the notion “to be better” for two services. In this study, two approaches for such evaluation are briefly sketched: the use of utility function and Pareto-dominance relationship. Utility function (Sun & Zhao, 2012; Lalanne, Cavalli, & Maag, 2012; Cardoso et al., 2004) is a computable function that maps the QoS vector into a single quality score $F(QoS)$. In other words, given two services $S_1$ and $S_2$ with the quality vectors $QoS_1$ and $QoS_2$, the service $S_1$ is better than $S_2$ if $F(QoS_1) > F(QoS_2)$. There are many ways for defining utility functions; the simplest option is to define such a function as a weighted sum of quality parameter values, $F(QoS) = \sum_{q_i \in QoS} w_i q_i$, where the main question is how to choose weights. In (Sun & Zhao, 2012), weights are chosen in such a way that a weighted parameter value is between 0 and 1 where 0 corresponds to the “worst” parameter value while ’1’ corresponds to the “best”. In (Xin et al., 2013), the weights are calculated based on a partial order relation between parameters. The “worst” and “best” parameter values essentially depend on a parameter. For instance, for the response time the less the value is, the better it is for the service quality, while for the availability or popularity a better value is the greater one.

The utility function is not always appropriate for the service quality comparison, especially for composite services. Last years, Pareto-dominance relationship, also often called skyline, is increasingly used (Xin et al., 2013; Shaoqian, Wanchun, & Jinjun, 2013). Once the relation “better” is set for the set of values of each parameter, the quality vector $QoS_1$ is said to dominate vector $QoS_2$ if all components of $QoS_1$ are “not worse” than corresponding components of $QoS_2$ and at least one component of $QoS_1$ is better than that of $QoS_2$. The importance of parameters may be easily taken into account, by considering the domination only over most important parameters. Less important parameters are not taken into consideration at all or the maximal discrepancy value can be set for these parameters.

Example. The vacation planner with the quality $QoS_1 = (20, 0.8, 0.3)$ skylines the service with the quality $QoS_2 = (25, 0.85, 0.3)$ when only the most important parameter, the response time, is taken into account. This domination is still valid if the maximum discrepancy value for the availability is 0.1. If all the parameters are considered equally important then neither $S_1$ nor $S_2$ skylines another service.

3.3 Deriving a Formal Specification

Web services published in repositories are often described using XML-based languages, such as WSDL, BPEL, WS-CDL (Web Service Choreography Description Language) depending on abstraction and hierarchical levels, etc. (Booth et al., 2004).

As web services often process permissible sequences of actions, finite state models are widely used for their analysis and synthesis. In (Rabanal, Rodriguez, Mateo, & Diaz, 2012), the authors extract a Finite State Machine (FSM) from the WS-CDL description using the tool DIEGO 2.0. In (Diaz et al., 2005), the BPEL and WS-CDL descriptions are translated into a system of communicating timed automata. In most papers, the extracted finite state models are used for automatic composition of services (Beek, Bucchiarone, & Gnesi, 2006) or testing and verification purposes (Tien-Dung, Felix, & Castanet, 2010; Honghao & Ying, 2011). As mentioned in (Héam, Kouchnarenko, & Voinot, 2010; Kondratyeva, et al., 2013b), finite state models can also be helpful for more precise quality evaluation.

Example. For the running example, the service finite automaton specification at a high abstraction level is given in Figure 1.

![Figure 1. An automaton for the vacation planner web service](image-url)
In both cases, the system moves to the final state (the same as the initial one).

An extracted formal model can be hierarchical. For example, if more detailed and precise description is needed, it is possible to unroll a composite state \( q_3 \) as shown in Figure 2.

![Automaton for representing a composite state](image)

When considering a state \( q_3 \) as a composite state the input \(?fstat\) is unrolled into a sequence of four inputs, namely, into inputs \(?class\), \(?travel\) time, \(?departure\) time and \(?price\). The input \(?class\) allows a user to specify the class of a requested flight, i.e., a business or an economy class, while the input \(?travel\) time refers to a maximal number of hours required for traveling. The input \(?departure\) time specifies the earliest time for a tourist to depart while the input \(?price\) is related to the maximal price of a plane ticket. Inputs \(?reserve\) and \(?not\ reserve\) correspond to the flight booking or canceling the order. The automaton outputs belong to the set \{?class, ?travel time, ?departure time, ?price, ?reserved\} and each input is followed by the same output. For instance, if a user specifies a preferable class by applying the ?class input the service outputs possible class options with the output ?class. If the class options do not satisfy the user he/she can apply the ?not\ reserve input and the system moves to state \( q_3^{\not{reserved}} \) where the output ?not\ reserved is produced. In other words, at state \( q_3 \) two outputs can be produced, i.e., the output ?date that corresponds to ?reserved output at state \( q_3^{reserved} \) or the output ?not\ reserved that corresponds to the same output produced at the state \( q_3^{\not{reserved}} \).

3.4 The Service Provisioning Step

The service provisioning step includes the analysis of available and required resources for the service under development. This issue involves business processes and related resources such as human-hours to be spent, the cost of implementation of a new service comparing to the usage of existing components, etc. All these parameters significantly influence the system architecture as well as a choice of component services and the implementation quality but their influence is rather implicit and for this reason, such business issues are left out of the scope of this paper. Nevertheless, knowing some network resources (server quality, internet speed, etc.) gives a chance for more precise evaluation of some service parameters such as the response time or availability of the service. The marketing analysis of people who are going to use such service can also refine the evaluation. The quality evaluation at this step can become a part of Service Level Agreement (SLA) (Keller & Ludwig, 2002). If the predicted quality is under desirable standards then the service requirements specification should be revised, i.e., a developer should come back to the previous development step.

Given a finite state model extracted from some service description, at the provisioning step the model can be refined by augmenting its states and/or transitions with weights, probabilities or other attributes based on the quality requirements.

In particular, in (Héam, Kouchnarenko, & Voinot, 2010), weighted automata are used for more precise service quality evaluation. The weight associated with each automaton transition represents the cost of the corresponding transaction execution (in terms of time, money, etc). The quality parameter values can be estimated via different execution paths of the corresponding automaton and some conclusions about the service quality can be drawn.

**Example.** In order to illustrate how the QoS of the vacation planner service can be evaluated at the provisioning step, the automaton in Figure 1 is augmented with weights for transitions that are responsible for service transactions. In this example, the weight is interpreted as a time needed for a corresponding service transaction, and a corresponding weighted automaton is shown in Figure 3. It is assumed to take at most 10 seconds for the vacation planner to check and to provide the available travel dates and not more than 3 seconds to confirm whether a ticket or a hotel room is reserved. The reservation of a ticket or a hotel room requires 10 seconds and thus, checking dates for a hotel room after booking the ticket lasts at most 13 seconds.

![Weighted automaton for the vacation planner service](image)
specification can be considered. The result is $10 + 3 + 1 + 13 + 3 + 6 = 36$ seconds. When using the formal model of a weighted automaton (Figure 3) the QoS can be evaluated more accurately by considering traces from the initial to the final state. In this case, as mentioned in (Heam, Kouchnarenko, & Voinot, 2010), the weight of each simple trace (without cycles) may be calculated and the longest transaction time corresponds to a trace with the maximal weight. In the example, this trace labels the sequence of states $q_0 q_1 q_2 q_3 q_4 q_5$ with total weight 29. Thus, the refined maximal transaction time for the vacation planner equals 29 seconds.  

4. SERVICE COMPOSITION AND IMPLEMENTATION

In this section, we briefly discuss the main aspects of deriving a quality-aware composite service. When a new service is derived by composing already existing web services, possibly developed by a third-party, two main questions arise: what service quality can be provided by existing components, and how to select components to achieve a higher quality of composite service. These two questions are considered in Section 4.2 and Section 4.3 correspondingly, while Section 4.1 presents the approach for a composite service architecture specification.

4.1 Basic Composition Patterns

For quality evaluation purposes, the functioning of the component services and the content of the messages they exchange are not considered. When designing the architecture of a composite service, task invocations between components can be expressed with a corresponding workflow and Figure 4 illustrates the basic composition patterns in which the workflow can be decomposed. When a component service is invoked, it executes some task (according to the composition requirements), and after completing the task, the service either produces the result if it is the final task, or invokes other components to execute further composition tasks. The same component service can be used for executing different composition tasks. To avoid any ambiguity further the execution of some task by the service is referred to as a component service.

The simplest workflow compositional pattern is sequential (Figure 4a) where the composite service is organized as follows: when the service $S_0$ completes a task then the service $S_1$ is invoked. In a conditional pattern (Figure 4b), also referred to as XOR-split pattern, $S_0$ invokes one and only one of services $S_1$, ..., $S_k$ depending on the results of the task execution. When probabilities are involved for each possible invocation, the equality $\sum_{i=0}^{k} p_i = 1$ must be held. When the service $S_0$ invokes several services $S_i$, ..., $S_k$ a parallel pattern (Figure 4c), or an AND-split pattern, is considered. Services executing tasks in parallel can be further merged in order to execute a required later task. If the next service is invoked only when all preceding services have completed their tasks, the synchronizing pattern (Figure 4d), or AND-joint pattern, is considered. Otherwise, if the next service is invoked after at least one service completes its task, a concurrent pattern (Figure 4e), or a XOR-joint pattern, is at hand. When some tasks should be repeated, the loop pattern (Figure 4f) is involved, and the number of repetitions may be either known a priori or can be calculated during the task execution. Without loss of generality, loops are considered to have a single starting point (service $S_1$ in Figure 4f) and any number of exits (service $S_k$ in Figure 4f). For each exit point, the probabilities of continuing the loop or of going out may be specified.

Example. The vacation planner service can be represented as the composition of two services: Flight Booking (FB) and Hotel Booking (HB) services. In the workflow in Figure 5, nodes S and F are initial and final nodes correspondingly.

**Figure 4.** Basic compositional patterns: (a) sequential, (b) conditional, (c) parallel, (d) synchronizing, (e) concurrent, (f) loop

**Figure 5.** Workflow for the vacation planner service

After asking for a flight ticket a user has three options: to ask for a hotel room when flight dates are set; to change flight dates or to quit the service when the flight dates cannot be changed for some reason. In the case of booking a hotel room, a request is processed only once. In both cases when a hotel room is booked or not booked, the service is quitted.
4.2 Aggregation Functions

Given a set \( \{S_1, S_2, \ldots, S_n\} \) of component services, their QoS vectors \( \{Q_1, Q_2, \ldots, Q_n\} \) (for instance, published by service providers or so-called service brokers (Zheng et al., 2013)), and the composition structure, the question is: what is the quality \( Q \) of the composite service?

For all basic patterns and their combinations the question of overall QoS evaluation has been studied properly and aggregation functions have been elaborated for a number of QoS parameters. The result of each aggregation function is the value of a corresponding attribute of the composite service. Some of these functions without considering probabilities for XOR-splits are summarized in Table 1. In Table 1, probabilities for XOR-splits and loops are taken into account when deriving aggregation functions. In these tables, the integer \( k \) denotes the number of involved services while the integer \( n \) is used for the number of loop iterations, and \( p_i \) is the probability of invoking the service \( S_i \).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Sequential</th>
<th>XOR-split</th>
<th>Parallel AND-join</th>
<th>Parallel XOR-join</th>
<th>Loop (n cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost ( (c) )</td>
<td>( \sum_{i=1}^{k} c_i )</td>
<td>( \sum_{i=1}^{k} c_i )</td>
<td>( \sum_{i=1}^{k} c_i )</td>
<td>( \sum_{i=1}^{k} c_i )</td>
<td>( \sum_{i=1}^{n} \sum_{i=1}^{k} c_i )</td>
</tr>
<tr>
<td>Availability ( (a) )</td>
<td>( \prod_{i=1}^{k} a_i )</td>
<td>( \prod_{i=1}^{k} a_i )</td>
<td>( \prod_{i=1}^{k} a_i )</td>
<td>( 1 - \prod_{i=1}^{k} (1 - a_i) )</td>
<td>( \left( \prod_{i=1}^{k} a_i \right)^n )</td>
</tr>
<tr>
<td>Response time ( (t) )</td>
<td>( \sum_{i=1}^{k} t_i )</td>
<td>max( (t_i) )</td>
<td>max( (t_i) )</td>
<td>min( (t_i) )</td>
<td>( \sum_{i=1}^{n} t_i )</td>
</tr>
<tr>
<td>Reliability ( (b) )</td>
<td>( \prod_{i=1}^{k} b_i )</td>
<td>( \prod_{i=1}^{k} b_i )</td>
<td>( \prod_{i=1}^{k} b_i )</td>
<td>( \prod_{i=1}^{k} b_i )</td>
<td>( \left( \prod_{i=1}^{k} b_i \right)^n )</td>
</tr>
</tbody>
</table>


4.3 Quality-Aware Component Service Selection

When the estimated quality of a composite service is unsatisfactory the question arises how the quality could be enhanced. According to the previous section, the quality of a composite service significantly depends on the composition structure and the quality of component services. The composition structure is mainly pre-determined by functional requirements and hence, cannot be easily changed. Correspondingly, in order to enhance the quality of a composite service there is an option of selecting a “better” component service, a so-called the quality-aware component selection problem. A “better” component service can be selected from a collection of services with similar functionalities based on a corresponding utility function or a skyline order relation.

The quality-aware component selection is a multi-dimensional optimization problem and often can be reduced to the well-known combinatorial problems such as the multi-choice knapsack problem (e.g., Xiaopeng & Chunxiao, 2009), the resource constraint project scheduling (Jaeger, Rojec-Goldmann, & Muhl, 2004), or the derivation of a shortest graph path under some constraints (Tao & Kwei-Jay, 2004), etc. Local and/or global approaches can be applied (Sun & Zhao, 2012; Xiaopeng & Chunxiao, 2009; Wu & Hong, 2010). Local selection approaches focus on choosing the best component for each task independently of other tasks, while the global optimization objective is to derive a composition with a better overall quality. For partially specified or derived on-the-fly compositions a local selection approach is reasonable, though, it cannot guarantee that the composite service satisfies the given quality requirements (Zeng, et al. 2003).

Sometimes workflows, which are often used for calculating the composition QoS, are insufficient for the precise evaluation of the quality of the composite service. Thus, more complex trace models such as finite transition systems are needed to accurately evaluate the quality of the composition when service component qualities are given.

Example. Correspondingly, using the above tables the QoS of the vacation planner can be calculated when knowing the QoS of FB and HB services.

http://hipore.com/ijsc
Global approaches which can ensure the composite service quality utilize the multi-dimensional optimization (Zeng, et al. 2003; Wu & Hong, 2010; Moser, Rosenberg, & Dustdar, 2012) and these approaches are rather computationally expensive. Novel promising approaches are rather the mixture of local and global approaches (Sun & Zhao, 2012; Xin, et al., 2013; Shaoqian, Wanchun, & Jinjun, 2013).

Example. For the running example, the QoS is the vector \( QoS = (t, r, a) \), and based on the workflow in Figure 5 and corresponding aggregation functions, the composite service response time and availability can be calculated. To calculate the popularity, though, the aggregation function should be specified. Based on the definition of the service popularity and regarding that the vacation is booked successfully only when both flight ticket and hotel room are booked, we conclude that for the vacation planner service the popularity could be calculated as the product of popularities of services FB and HB. Consider three different hotel booking services HB1, HB2, HB3 with the QoS vectors \( QoS_1 = (18, 0.5, 0.9) \), \( QoS_2 = (20, 0.7, 0.9) \), and \( QoS_3 = (10, 0.8, 0.6) \), correspondingly. If the flight booking service FB has the QoS vector \( QoS_F = (15, 0.8, 0.9) \), then for the corresponding composite services, it holds that \( QoS_{C1} = (63, 0.4, 0.65) \), \( QoS_{C2} = (65, 0.56, 0.65) \), and \( QoS_{C3} = (55, 0.64, 0.44) \). As stated above, the popularity and the availability are assumed to be less important than the response time. Since all solutions comply the restriction that the popularity is at least 0.2, the service HB3 can be chosen as a component service due to the best response time.

4.4 The Implementation Step

When all decisions are taken about the service hierarchy and components, the non-existing component services and the overall composition are implemented. At this step, formal descriptions using finite state models can essentially help as there are many tools which allow automatic code generation (see, for example, Mtsweni, 2012). A developed implementation is then verified and tested for checking that the implementation conforms to its specification (Tien-Dung, Felix, & Castanet, 2010; Lallali, et al., 2008; Honghao & Ying, 2011). Other properties of the implementation such as security and robustness can be also tested at this level. Nevertheless, the quality related features which are expected according to the service specification usually are tested at the next step when the service is started to be used. If some parameter values do not satisfy a developer he/she can come back to the previous steps in order to redesign the service.

5. Service Usage and Management

The main objectives of service management include, but are not restricted to, improving service quality and ensuring that the service satisfies functional and non-functional requirements (Booth et al., 2004; IBM, 2001). Requirements that service should meet can be specified in the Service Level Agreement (Keller & Ludwig, 2002) and are checked via service monitoring when various data are collected and analyzed. At this step, detecting the SLA violations becomes one of the main service management objectives. Functional violations are out of the scope of this paper; in this section, we only discuss how the service effectiveness can be evaluated during the step of usage and management. The data collected via monitoring of service activities fail to completely illustrate the service profitability, since the main purpose of each service is to satisfy an end-user. Still, there is no confidence that a user is satisfied even when the QoS is rather high. For this reason, at the management step the QoE value is involved. It should be noted that the QoE can also be automatically evaluated using finite state models.

Section 5.1 contains a short discussion on the service usage monitoring, since this monitoring is a source for the service parameters’ information and includes a short discussion on how the QoE can be evaluated based on the collected data. In Section 5.2, the automated evaluation of

![Table 2. Aggregation functions for quality parameters when probabilities are involved](http://hipore.com/ijsc)
the QoE is discussed, since a direct feedback from users is not always available.

5.1 Traffic Monitoring for Quality Evaluation

Based on traffic analysis some statistics can be collected, such as the maximal time delay between transactions, the probability of losing a package, bandwidth etc. (Khirman & Henriksen, 2002). It is possible to consider how these parameters influence the traffic QoS evaluation. However, the above parameters are more related to a transport level than to the service under investigation itself. The choice of parameters to be monitored during the service usage is important for further management and concluding how each parameter influences the end-user satisfaction and how the service can be modified in order to improve its quality.

The main problem for the QoE evaluation is that this metric is highly subjective. Therefore, the most common and straightforward way to evaluate the user satisfaction is to use experts or directly ask a user through a questionnaire. Questions in a questionnaire can widely vary depending on which key parameters a service provider is interested in: whether a user has a confidence in the service, or whether a user is going to use the service again and recommend it to his/her friend, etc. (Dale et al., 2007). The questionnaire is then analyzed and a formal model for the QoE computing/prediction can be developed. Based on this model decisions could be made for which groups of users the service can be attractive and which business profit can be obtained. Another issue can be concerned about why the service is not attractive for some groups of users and what should be modified in order to make it more attractive.

Example. For the running example, during the service usage two parameters are monitored: the overall response time and the service popularity. The QoE score is supposed to be evaluated through some appropriate feedback procedure. The response time is an integer and is measured in seconds while service popularity is a rational between 0 and 1. The QoE score is an integer between 1 (the worst score) and 5 (the best score). The corresponding sample is presented by Table 3.

<table>
<thead>
<tr>
<th>Response time (sec)</th>
<th>Popularity</th>
<th>QoE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>0.8</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>0.4</td>
<td>2</td>
</tr>
</tbody>
</table>

5.2 State Model Based QoE Evaluation

Having a statistical correspondence between QoS values and QoE scores after a number of experiments, one may derive complex approximation functions for automatic calculation of QoE based on the known QoS value (see, for example, Lalanne, Cavalli, & Maag, 2012). Analyzing those approximation functions a provider can estimate which parameters are crucial for the QoE evaluation. Unfortunately, it is not always possible to derive approximation functions, and thus, more complex analysis is required for automated QoE prediction.

Though the question how the human brain works still remains unsolved, advanced studies of cognitive processes can help to elaborate adequate models of human perception and decision making procedures. Human behavior being nondeterministic or even random can hardly be reduced to some formal algorithms. Therefore, self-adaptive models that can be trained are promising for user experience evaluation. Some of self-adapting models can be considered as finite state models and almost all of them rely on pre/post conditions that can be expressed in terms of IF-THEN operator such as “if the service works fast I am rather happy with this service” (Kondratyeva et al., 2013b). Artificial intelligence techniques, such as fuzzy logic and neural networks seem to be promising for establishing correspondence between measured objective quality parameters and user satisfaction grades. For example, in (Pernici & Siadat, 2011; Pokhrel et al., 2013) fuzzy logic is used to calculate the correlation between network parameters and the QoE value. Since this correlation differs from service to service and is usually significantly nonlinear, it is essential to introduce adaptive algorithms to derive function coefficients automatically for given services. In (Pernici & Siadat, 2011) neural network learning algorithms are used to develop the fuzzy logic based tool to determine values and grades (excellent, good, average, or poor) for services. Talking about finite state models, a decision tree can be efficiently used for the QoE evaluation/prediction, as it is illustrated for the running example. Each node of a decision tree represents corresponding service parameter that is evaluated at a tree level, each transition corresponds to the possible parameter value.

Example. In the running example, corresponding decision tree has three levels and the root is labeled by the most important quality parameter, i.e., the response time (t). The second level of the tree contains nodes that are related to the service popularity (r). Terminal nodes of the tree are labeled by corresponding QoE values (Figure 6).

![Figure 6. The decision tree for the vacation planner service](http://hipore.com/ijsc)
time equals 10 possible subtree branches lead to terminal nodes with the QoE that is at least 3.

When new statistical data are gathered the decision tree should be adapted to these data by adding new branches. When predicting the QoE for parameter values which do not label a path from the root to a terminal node, additional rules should be set. For example, a branch with minimal discrepancy in parameter values can be selected for the QoE estimation. A decision tree can also be used for the analysis of the parameter importance for the users’ satisfaction based on the difference in the QoE score between neighbor subtrees. Readers may refer to (Kondratyeva et al., 2013b) in order to find out more about other finite state models that are often used for the automatic prediction of the QoE value.

6. CONCLUSIONS

In this paper, the advantages of using finite state models for the web service automatic QoS and QoE evaluation have been discussed. At each service development step, an appropriate finite state model has been considered for refining the quality evaluation. The running example of the vacation planner service clearly illustrates that finite state models relying on input/output sequences of service specification improve the accuracy of the QoS evaluation.

Since the user satisfaction is a priority of each web service, finite state models for the QoE evaluation that can be derived at the usage and management step have been briefly discussed. However, the development of new adequate and scalable formal models for the automatic QoE evaluation remains a challenging problem. Moreover, other neoteric quality metrics such as Quality of Design or Quality of Business are left out of the scope of this paper and need additional research.

7. ACKNOWLEDGMENT

The work is partly supported by ANR PIMI project (France). Olga Kondratyeva gratefully acknowledges the support of Futur et Ruptures scholarship from Fondation Telecom (France) and a scholarship of the President of Russian Federation for gifted students.

8. REFERENCES


Cubera, F., Dufller, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 6(2), 86-93, doi: 10.1109/4236.991449


Authors

Olga Kondratyeva has received her Master degree in Radiophysics (profile: Information systems and technologies) from Tomsk State University, Russia, in 2012. Currently she is a PhD student at double-degree program between Telecom SudParis, France and Tomsk State University, Russia. Her research interests include automata theory, service quality evaluation, timed-constrained systems, communicating systems.

Natalia Kushik has received her diploma degree in Applied Mathematics from Tomsk State University, Russia, in 2010. Shehas got a PhD degree in 2013 and worked as a PostDoc at Telecom SudParis, France. Her research interests include automata theory,
service quality evaluation, software testing and verification.

Professor Dr. Ana Rosa Cavalli is professor at Telecom SudParis and the head of Software and Networks department since 1990. Her research interests are on specification and verification, testing methodologies for conformance and interoperability testing, active testing and monitoring techniques, the validation of security properties and their application to services and protocols. She has had more than 200 peer review papers published in international conferences and journals. She led and participated recently in numerous national (CARRIOCAS, EXOTICUS, POLITESS, E-COMPAGNON) and European projects (ASK IT, E-LANE, SHIELDS, IPNQIS, DIAMONDS, INTER-TRUST) contributing to the modelling, monitoring, test, QoS and QoE analysis. She is member of several Steering and Program Committees of numerous international conferences: IEEE ICNP, IEEE ICST, IFIP CFIP, SARSII, AMOST-T, SETOP, IEEE ICSTSS, QESS. She has been co-chair of TESTCOM'95, SDL'97, PSTV/FORET'98, ICNP2002 et CFIP'2003. She has been domain chair of ICNP 2009, co-chair of the IEEE ICST 2010 and SETOP 2010 conferences.

Professor Nina Yevtushenko received her diploma degree in Electrical Engineering from Tomsk State University. Until 1991 she worked as a researcher with the Siberian Scientific Institute of Physics and Technology. From 1991 she joined Tomsk State University as a professor and presently she leads a research team working on the synthesis and analysis of discrete event systems. She also stayed as a visiting researcher/professor at Moscow State University, Université de Montréal, University of Ottawa, TelecomSudParis (former: Institute National des Telecommunications), Verona University. She published four books and many research papers. Her research interests include formal methods, automata theory, distributed systems, protocol and software testing.
AN EVENT-BASED APPROACH FOR DECLARATIVE, INTEGRATED AND SELF-HEALING WEB SERVICES COMPOSITION

Ehtesham Zahoor*, Kashif Munir*, Olivier Perrin** and Claude Godart**

*National University of Computer and Emerging Sciences
A.K. Brohi Road, Sector H-11/4, Islamabad, Pakistan
ehtesham.zahoor@nu.edu.pk, kashif.munir@nu.edu.pk

**Université de Lorraine, LORIA
BP 239 54506 Vandoeuvre-lès-Nancy Cedex, France
olivier.perrin@loria.fr, claude.godart@loria.fr

Abstract

Web services are defined to be the software systems that provide interoperable machine-to-machine interaction over a network. Individual services may need to be composed and the composition process design, verification and monitoring are thus active and widely studied research directions. However, the traditional approaches are both procedural (and rigid) and do not address the need of integrating these related dimensions using a unified formalism.

In this paper, we propose an event-oriented framework called DISC that is both declarative and serves as a unified framework to bridge the gap between the process design, verification and monitoring. It provides a flexible and highly expressive composition design that can accommodate various aspects such as data relationships and constraints, Web services dynamic binding compliance regulations, security or temporal requirements etc. Furthermore, the DISC framework allows for instantiating and verifying the composition design and for monitoring the process while in execution.

Keywords: SOA, Web services composition, Declarative, Integrated, Verification, Event-calculus, Self-healing

1. INTRODUCTION

Web services are in the mainstream of information technology and are paving way for inter and intra organizational application integration. High-level languages such as WS-BPEL and specifications such as WS-CDL and WS-Coordination extend the service concept by providing a method of defining and supporting orchestration (composition) of fine-grained services into more coarse-grained value added processes. The Web services composition process has different life-cycle stages. First, the process designer needs to model the composition process by using the fine-grained services to define new added-value processes. Then, the composition process needs to be verified to identify any anomalies and conflicts (such as deadlocks) in the process specification before execution. However, as the Web services are autonomous and only expose their interfaces, composition process is based on design level service contracts and the actual execution of composition process may result in the violation of the design-level services contracts due to errors such as network or service failures, change in implementation or other unforeseen situation etc. This highlights the need to monitor and detect the errors and accordingly react to cater for them.

Web services composition is a highly active research direction and in the literature, a number of approaches have been proposed that handle different stages of process life cycle. However, the traditional approaches have two major short-comings; first these approaches focus only on some stages of process lifecycle and this lack of integration results in a complex model such as mapping the BPEL based process specification to a particular automata with guards and using SPIN model checker for verification; BPEL to timed automata and using UPPAAAL model checker (Guermouche & Godart, 2009b) for checking temporal properties. The lack of integration leads to the approaches that do not allow to learn from the runtime failures to provide the recovery actions, such as re-planning or alternative path finding, to recover from the monitored runtime violations based on current state of the process. Further, the proposed approaches aim to build on top of the traditional approaches (such as BPEL, OWL-S) which focus on the control flow of the composition using a procedural approach and as pointed out in (van der Aalst & Pesic, 2006), they over constrain the composition process making it rigid and not able to handle the dynamically changing situations. Furthermore, the focus on data, temporal, security aspects and other nonfunctional requirements is not thoroughly investigated. Adding these aspects makes the process very complex, as again they are proposed as a new layer on the top of existing BPEL based processes. The proliferation of partial solutions, the lack of expressiveness and simplicity to handle both functional and non-functional aspects, the lack of integration, the lack of recovery actions and the lack of flexibility mark the motivation for our work.

In this paper, we propose a declarative event-oriented framework, called DISC (Declarative Integrated Self-healing web services Composition), that serves as a unified framework to bridge the gap between the process design,
verification and monitoring and thus allowing for self-healing Web services composition, Figure 1. Specifically, our contributions include:

- **Expressive composition design:** The proposed framework allows for an EC based composition design that can accommodate various aspects such as partial or complete process choreography and exceptions, different control flow constructs, data relationships and constraints, Web services dynamic binding, compliance regulations and other non-functional (such as different temporal and security) aspects.

- **Extensible approach:** The EC allows for integrating the existing work on composition design (Cicekli & Yildirim, 2000), authorization (Bandara, Lupu, & Russo, 2003; Gaaloul, Zahoor, Charoy, & Godart, 2010), and work on modeling other related aspects. Different EC models, presented in this work, are organized into independent generic patterns that can be added to EC based process specification facilitating reuse and extension.

- **SAT-based process verification:** We advocate the need for the satisfiability solving for the verification of declarative processes. We have also proposed filtering criteria, based on the proposed patterns and on the structure of conflict clauses that can help to identify the clauses of interest. The proposed filtering approach is generic and can be applied to any problem specified using EC.

- **Event-based monitoring:** The proposed monitoring approach is built upon an event-based declarative composition design and this result in an integrated approach that allows to reason about the events. It does not require defining and extracting events from process specification, as the events are first class objects of both design and monitoring frameworks.

- **Implementation support:** The generic pattern-based approach allows us to implement a Java-based application, called ECWS, which allows abstracting the EC models from the process-designer and automates the composition process specification and verification. We have modified the DECReasoner code to gain substantial performance improvement as evident in performance evaluation results.

2. **Motivation and Related Work**

The motivation for our work originates from the need for process modeling, analysis and monitoring in a crisis situation. A crisis situation is highly dynamic. It demands for a process that is possibly partially defined. It is characterized by temporal and security constraints and uncertainty, and multiple and possibly changing goals. Hence, it requires the composition process to be more flexible to adapt to a continuously evolving environment. The crisis scenario brings together two related dimensions of organization and situation measurement. The organization dimension encompasses the design-time composition process modeling. Most of the proposed approaches for the composition design can be divided into Workflow composition and AI planning based approaches.

A problem of traditional Workflow oriented approaches (such as WS-BPEL and WS-CDL) is that they over-constrain the process that must be specified with the exact and complete sequence of activities to be executed. Although, this adds a lot to the control over the composition process, this control comes at the expense of process flexibility, making the process rigid to adapt to continuously changing situations, a detailed discussion can be found in (Pesic & van der Aalst, 2006). Furthermore, the traditional approaches make it difficult to model complex orchestrations, i.e. those in which we need to express not only functional but also non-functional requirements such as cardinality constraints (one or more execution), existence constraints, negative relationships between services, temporal and security requirements on services. Some declarative approaches such as (van der Aalst & Pesic, 2006; Quartel et al., 2007) allow for defining process in a flexible way, but our approach allows for the same set of constraints to be used not only for composition modeling and analysis, but also for monitoring and violations feedback to composition design.

The design time verification of the composition process before execution is also an important aspect and a large number of proposed approaches require the mapping of composition process to some formal logic to be then used for verification. These approaches include mapping the BPEL process to particular automata with guards, and using SPIN model checker (Fu et al., 2004), BPEL to timed automata and using UPPAAL model checker (Guerrouche & Godart, 2009b), BPMN constructs into labeled petri-nets then using BPEL2PLML (Dijkman, Dumas, & Ouyang, 2007) and others. Some approaches do consider the prospect of building upon a composition design that can be verified, such as Petri-net based design and verification as proposed in (Yi & Kochut, 2004), the process algebra based approach to compose and verify the composition process and the restricted abstract BPEL process to analyze the correctness.
The second dimension a crisis situation focuses on is the situation measurement. The crisis handling composition process should be able to measure and adapt to continuously changing situation. This leads to the problem of Web services monitoring (Barbon, Traverso, Pistore, & Trainotti, 2006; Baresi, Guinea, Pistore, & Trainotti, 2009; Mahbub). The problem to effectively monitor and recover from the anomalies during process execution spans different related domains. Workflow management systems, in general, rely on an exception handling approach (Russell, van der Aalst, & ter Hofstede, 2006; Vanhatalo, Volzer, Leymann, & Moser, 2008). For the self-healing systems a number of approaches have been proposed (Ghosh, Sharman, Rao, & Upadhyaya, 2007; Griffith, Kaiser, & Lopez, 2009). In this paper, we would consider monitoring and recovery mechanisms for only services based processes, a detailed discussion about monitoring and recovery in different systems can be found in (Friedrich, Fugini, Mussi, Pernici, & Tagni, 2010).

Traditional composition process monitoring approaches, such as BPEL, build upon the exception handling approach for errors handling and allows to define different types of exception handlers and corresponding actions based on process state. However, it may not be always possible to foresee errors and specify the exceptions at the design-time. In the literature, a number of composition process monitoring approaches have been proposed, but in general, they are proposed as an extension to some particular runtime and are tightly coupled and limited to it (Beeri, Eyal, Milo, & Pilberg, 2008; Baresi, Guinea, Nano, & Spanoudakis, 2010; Sun, Li, & Zhang, 2009; Wu, Wei, & Huang, 2008). As a result, they do not consider other sub-systems and processes that can be used for monitoring (Moser, Rosenberg, & Dustdar, 2010).

In addition, the traditional monitoring approaches (Barbon et al., 2006; Baresi et al., 2009; Moser et al., 2010; Ardagna, Comuzzi, Mussi, Pernici, & Plebani, 2007; Friedrich et al., 2010) build upon composition frameworks that are highly procedural, such as BPEL, and this in-turn poses two major limitations. Firstly, they limit the benefits of any event-based monitoring approach as the events are not part of the composition framework and functional and non-functional properties are not expressed in terms of events and their effects. Secondly, the use of procedural approach for process specification does not bridge the gap between organization and situation in a way that it is very difficult to learn from run-time violations and to change the process instance (or more importantly process model) at execution time, and it does not allow for a reasoning approach allowing for effects calculation and recovery actions such as re-planning or alternate path finding as we discussed in (Zahoor, Perrin, & Godart, 2010a). Our approach builds on an event-based framework and events are first class objects in both composition design and monitoring framework. This allows to reason about events during execution and allows for effects calculation and different types of recovery actions. A more detailed discussion can be found in (Zahoor E., 2011).

The growing need for incorporating the security and temporal aspects in the services composition has led to many approaches that aim to handle security and temporal aspects in the services composition at different levels of process lifecycle. In general, the proposed approaches are either based on the procedural approaches such as BPEL, lack expressiveness and ease to model complex temporal and security aspects (or most importantly, their combination) or mostly focus on only part of the problem. The proposed approaches for incorporating temporal aspects include (Guermouche & Godart, 2009a; Ponge, Benattallah, Casati, & Tournati, 2007) however, these approaches do not address the need for a unified framework and only focus on part of the problem. The proposed approaches also include (Kazhamiakin, Pandya, & Pistore, 2006), in which authors introduced a formalism called WSTTS to capture timed behavior of Web services and then using this formalism for model-checking WS-BPEL processes. The approaches also include ISDL (Quartel, Dijkstra, & van Sinderen, 2004), which uses time attributes to represent properties. In order to verify the timed properties authors proposed converting the BPEL process specification to timed automata and using UPPAAL model checker (Guermouche & Godart, 2009b). There have been many approaches that aim to handle the security aspects in the Web services composition (Menzel, Thomas, & Meinel, 2009; Garcia & de Toledo, 2008; Souza et al., 2009) however, as similar to the approaches for incorporating the temporal aspects, they only focus on a part of the problem.

The approaches that deal with the representation of the security aspects and aim to incorporate the security requirements into the business process definition include (Menzel et al., 2009; Neubauer & Heurix, 2008; Rodriguez, Fernandez-Medina, & Piattini, 2007). Furthermore, there have been approaches that aim to incorporate security requirements in the executable composition (Garcia & de Toledo, 2008; Chollet & Lalanda, 2008) or their enforcement at execution time (Song, Sun, Yin, & Zheng, 2006; Menzel et al., 2009). In (Souza et al., 2009), the authors proposed to use a formalism that allows for incorporating security aspects at different levels of abstraction, however the approach is procedural as it is based on and extends BPMN notations, lacks formal representation and does not allow for verification of and reasoning about the security properties.

The proliferation of partial solutions, the lack of expressiveness and simplicity to handle both functional and non-functional aspects, the lack of integration, the lack of recovery actions and the lack of flexibility mark the motivation for our work.
3. COMPOSITION DESIGN

The composition process modeling is the first and most important stage of the composition process life cycle, Figure 1. The objective of composition process modeling is to provide high-level specification independent from its implementation that should be easily understandable by the process modeler who creates the process, the developers responsible for implementing the process, and the business managers who monitor and manage the process. The proposed DISC framework allows for a composition design that can accommodate various aspects such as partial or complete process choreography and exceptions, data relationships and constraints, Web services dynamic binding, compliance regulations, security and temporal requirements or other non-functional aspects. The composition design (and so are the other phases of the proposed framework) is based on EC.

3.1 EVENT CALCULUS (EC).

In order to model the composition design, our approach relies on the EC (Kowalski & Sergot, 1986). EC is a logic programming formalism for representing events and their effects and can infer “what is true when” given “what happens when” and “what actions do”, Figure 2. The “what is true when” part both represents the state of the world called initial situation and the objective or goal. The “what actions do” part states the effects of the actions. The “what happens when” part is a narrative of events. A detailed presentation can be found in (Zahoor et al., 2009).

The choice of EC is motivated by several reasons. First, EC integrates an explicit time structure independent of any sequence of events (possibly concurrent). Then, given the composition design specified in the EC, a reasoner can be used to instantiate the composition design. EC is very interesting as the same logical representation can be used for verification at both design time (static analysis) and runtime (dynamic analysis and monitoring). Further, it allows for a number of reasoning tasks that can be broadly categorized into deductive, abductive, and inductive tasks. In reference to our proposal, at composition design stage “abduction reasoning” can be used to find a set of plans or to identify any conflicts and at the composition monitoring stage, “deduction reasoning” can be used to calculate the effect of run-time violations. Figure 3 shows the mapping of EC to the proposed framework.

What happens when | Specified initial orderings (partial plan, if any) and the sought execution plan for the specified goal
What actions do | Actions such as service invocation, nodes binding and associated effects
What is true when | The composition design including the Initial situation such as constraints, dependencies and the goal for the composition process (If any)

Figure 3. EC for the proposed framework

The EC models are presented using the discrete EC language (Mueller, 2006). We will only present the simplified models that represent the core aspects, intentionally leaving out the supporting axioms. All the variables (such as service, time, node etc) are universally quantified and in the case of existential quantification, it is represented with variable name within curly brackets, {variablename}. For spacing issues, we abbreviate Response to Resp and Service to Serv.

3.2 COMPONENTS.

The various components that constitute the composition design can be broadly divided into activity and service categories. Activity is a general term for any work being performed. The services include either the Web services instances already known or abstract Web services (called nodes) that need to be instantiated (discovered) based on some specified constraints.

Activities. Each activity can have an activity life cycle as it changes states from being started till its completion. In order to model the activities using EC we can define events that
represent the actions required to start and finish the activities and defining EC fluents can represent the activity state. The EC based model for representing activities is shown in Figure 4.

```
sort activity
fluent Started(activity), Finished(activity)
event Start(activity), End(activity).
Initiates(Start(activity), Started(activity), time).
Initiates(End(activity), Finished(activity), time).
Terminates(End(activity), Started(activity), time).
Happens(End(activity), time) → HoldsAt(Started(activity), time).
HoldsAt(Started(activity), time) → !Happens(Start(activity), time).
HoldsAt(Finished(activity), time) → !Happens(End(activity), time).
HoldsAt(Finished(activity), time) → !Happens(Start(activity), time).
!HoldsAt(Started(activity), 0).
!HoldsAt(Finished(activity), 0).
```

Figure 4. Activities model (with states)

In the model as shown in Figure 4, we first define an EC sort named activity. The instances of the sort represent the actual activities. Then, we define events that represent the actions to change activity state and fluents that represent the activity state. An activity state can either be Started or Finished and the events that are responsible for state change are the Start and End events. Then, in the EC model above, we define the Initiates axioms that specify that if the Start event happens at some time, the fluent Started holds true after that time and thus the Initiates axioms represent the state change. As a result of End event, the activity state changes to Finished (represented by the second Initiates axiom). We also define a Terminates axiom, which specifies that as a result of End event, the activity state is no longer Started (and thus the fluent Started does not hold).

We define some axioms to control the invocation of specified events such as the End event should only Happen once the activity has already been started, and the fluent Started holds. Similarly, other axioms specify that once the activity has started (or finished) the Start (and End) events should not Happen. Finally, the last two axioms specify that the initial condition for the fluents that they do not hold at time point 0. In this work, we model different kind of activities that include:
- Activities without intermediate states.
- Activities that may require restart - possibly within loop body.
- Activity model for dynamic task delegation.

Space limitations restrict us to detail the EC models for different kind of activities. However, a detailed discussion can be found in (Zahoor E., 2011). Let us briefly discuss how such an EC based composition model can be used for reasoning by defining two instances (ActivityA and ActivityB) of sort activity. In the EC model shown in Figure 5, we first import the EC core files (root.e and ec.e) and then we import the activity.e file, which contains the EC model for activities modeling shown earlier.

Then, we define instances of sort activity that represent the activities and also the goal for the process that is to have the activities Finished (and thus requiring the fluent Finished to hold) at time-point 2. Finally, we specify the range for time/offset and any options for the DECReasoner, in this case requiring not to show predicates (showpred off).

```
;including helper files
load foundations/Root.e
load foundations/EC.e
load includes/activity.e

;creating instances representing activities
activity ActivityA, ActivityB

;initial conditions for the fluents
!HoldsAt(Finished(activity), 0).

;composition goal
HoldsAt(Finished(activity), 2).

range time 0 2
range offset 1 1
option showpred off
```

Figure 5. Activities definition

Web Services. The proposed composition design also allows modeling the Web services. As the proposed framework aims to reason about the composition process, we only model the core aspects leaving out the details, which are only needed for services execution. However, proposed models can be extended to handle other aspects, if needed. Different Web services models have been proposed, that include:
- Web services with different synchronization modes (synchronous / pull-push based asynchronous).
- Web services that require re-invocation - possibly within loop body.
- Web services to be discovered based on constraints, called Nodes.

The event-calculus model for the synchronous Web services is shown in Figure 6. Space limitations restrict us to detail different EC models and a detailed discussion can be found in (Zahoor E., 2011).
3.3 CONTROL/DATA FLOW SPECIFICATION.

We have proposed the EC based patterns for specifying different control/data flow constructs, such as Dependency, Split/Join and different Split/Join Schemes, Conditional invocation/start of components; Iteration and data flow between different components. Different constructs that are modeled include:

- Dependency, enforcing invocation order.
- Split construct with different split-schemes (parallel/alternative/exclusive).
- Join construct with different join modes (all/exactly-one/at-least-one/subset).
- Conditions and conditional invocation of components.
- Iteration, components re-invocation unless loop exit condition is false.
- Request/Response data associated with and Message flow between components.

The dependency construct specifies the control and/or data flow dependency between different components and requires that the dependent component should not be started/invoked unless the component on which it is dependent, is completed/response data has been received, Figure 7. In order to model the dependency between different components using event-calculus we can add the axioms to the process specification that specify that the initialization/start event of the dependent component should not happen unless the fluent representing the completion of the source component holds. The EC based generic model for the specification of dependency construct is shown in Figure 8.

\[
\text{Happens(DEPENDENT\_EVENT, time)} \rightarrow \\
\text{HoldsAt(SOURCE\_FLUENT, time)}.
\]

Figure 8. Pattern for specifying dependency construct

In order to use the pattern, we need to update the source and target events. The DEPENDENT\_EVENT should be substituted by the start/invocation event for the dependent component while the SOURCE\_FLUENT should be substituted by the fluent representing the completion/response reception of the component on which other is dependent. An intuitive representation of dependency construct for specifying that some activity, DependentActivity, is dependent on the completion (response data received) of a service called SomeService is shown in Figure 9.

![Figure 9. The dependency construct – usage example](image)

One important advantage of the proposed approach is that it allows defining events-based dependency and thus dependency can be specified between any two events. This in-turn makes it possible to specify dependency not only on the successful completion of a component but also on the partial state of a component. Examples include to define the dependency on a activity being started, in execution or on a service being invoked (and not yet completed) or on the data, such as data has been received, expired or the reception of some particular data values.

Space limitations restrict us to detail EC models for other control and data flow constructs. A detailed discussion can be found in (Zahoor E., 2011).

3.4 MODELING NON-FUNCTIONAL ASPECTS.

The use of EC as modeling formalism also allows modeling different non-functional (temporal and security) aspects. Different patterns for temporal aspects include:

- Response time
- Restart/Refresh
- Invocation time-frame and delay
- Allen’s interval algebra
- Modeling time-units

Here, we only discuss the EC based model to specify the Allen’s temporal relationships. Allen’s Interval Algebra is a calculus for temporal reasoning that was introduced by James F. Allen in 1983. The calculus defines possible relations between time intervals and provides a composition
table that can be used as a basis for reasoning about temporal descriptions of events.

<table>
<thead>
<tr>
<th>Base relation</th>
<th>Representation</th>
<th>Event-calculus modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp(_A) before Comp(_B)</td>
<td>Component A, Component B</td>
<td>Specify DELAY between Comp(_A) completion and Comp(_B) invocation</td>
</tr>
<tr>
<td>Comp(_A) meets Comp(_B)</td>
<td>Component A, Component B</td>
<td>No DELAY between Comp(_A) completion and Comp(_B) invocation</td>
</tr>
<tr>
<td>Comp(_A) overlaps Comp(_B)</td>
<td>Component A, Component B</td>
<td>Specify OVERLAP_DELAY, time in which Comp(_A) and Comp(_B) are in concurrent execution</td>
</tr>
</tbody>
</table>

**Figure 10. Base relations for Allen’s Interval Algebra**

The base relations for the Allen’s interval algebra can be mapped and applied to the proposed framework, as shown in Figure 10. We briefly present the patterns for specifying after/meets/overlaps relations using EC in Figure 11. A more detailed discussion about other relationships can be found in (Zahoor E., 2011).

COMP\(_A\) after COMP\(_B\)
Happens(EVENT\(_B\), time) → Happens(EVENT\(_A\), time+DELAY).
Happens(EVENT\(_A\), time) → Happens(EVENT\(_B\), time).

COMP\(_A\) meets COMP\(_B\)
Happens(EVENT\(_B\), time) → Happens(EVENT\(_A\), time).
Happens(EVENT\(_A\), time) → Happens(EVENT\(_B\), time).

COMP\(_A\) overlaps COMP\(_B\)
Happens(EVENT\(_B\), time) → Happens(EVENT\(_A\), time+OVERLAP_DELAY).
Happens(EVENT\(_A\), time) → Happens(EVENT\(_B\), time+OVERLAP_DELAY).

**Figure 11. EC patterns for Allen’s interval algebra**

### 4. Process Instantiation and Verification

Once the EC based composition design has been specified, an EC reasoner can be used to instantiate the composition process to find a solution respecting all the functional and non-functional constraints associated with the process. The solution returned by the reasoner states what events happen at which time-points and also shows the effects those events have on the fluents, and the instantiated solution serves as a plan for process execution. The process instantiation phase may result in a number of solutions in the case of loosely constrained process. A particular solution from the set is chosen for execution based on either user-choice or based on some criteria such as overall-cost etc. In reference to temporal properties, one criterion for solution-selection is minimal time requiring to find a solution specifying to complete the execution process in minimal possible time.

As an example, consider the basic EC based model, Figure 5, as shown earlier in the composition design phase. Invoking the EC reasoner for the model gives us the solution shown as shown in Figure 12.

![Discrete EC Reasoner 1.0](loading activity instances.e)
loading foundations/Root.e
loading foundations/EC.e
loading includes/activity.e
32 variables and 78 clauses relsat solver
1 model
—
model 1:
0
Happens(Start(ActivityA), 0). Happens(Start(ActivityB), 0).
1
+Started(ActivityA). +Started(ActivityB).
Happens(End(ActivityA), 1). Happens(End(ActivityB), 1).
2
-Started(ActivityA).
-Started(ActivityB). +Finished(ActivityA).
+Finished(ActivityB).

**Figure 12. Solution returned by the reasoner**

The solution returned by the reasoner shows that if the Start events happen (representing that the activities are thus being started) at time-point 0, the fluents Started hold at time-point 1 as indicated by the + sign shown next to them at time-point 1 (representing that the activities state has been changed to Started). Once the activities have been started, the End events happen at time-point 1 to have the fluents Finished hold at time-point 2, that was the specified process goal. Note that the End events also make the fluents Started does not hold as indicated by the − sign shown next to them at time-point 2, representing the activity state is no longer Started.

The proposed approach for the verification of declarative Web services composition processes allows for both model checking the verification properties and for identifying and resolving the conflicts in the process specifications a result of process verification. The approach is based on satisfiability solving and requires verification properties to be added to the process specification and then encoding the problem into a SAT problem and then using the SAT-solver to find a solution or a set of unsatisfied clauses. The verification properties are added to the process specification and in general they can either be based on

http://hipore.com/ijsc
occurrence (or absence) of some specific event, as represented by EC events, or they can be based on satisfaction (or dissatisfaction) of some property, as represented by EC fluents. Figure 13 shows the general structure of EC axioms for the specification of verification properties.

\[
(!) \text{Happens}(\text{SomeEvent}(), \text{SomeTimePoint}) \rightarrow \text{Happens}(\text{SomeOtherEvent}(), \text{SomeTimePoint}).
\]

\[
(!) \text{HoldsAt}(\text{SomeFluent}(), \text{SomeTimePoint}).
\]

Figure 13. Verification properties specification

The EC to SAT encoding can be very large especially with the increase in time-points/free variables in axioms and with the complexity of the composition process. As a result, the set of un-satisfiable clauses (termed as unsatisfiable core) can be very large. We thus propose to filter the unsatisfiable core and to only consider the encoded clauses of interest. Space limitations restrict us to detail the filtering process, a detailed presentation and example can be found in (Zahoor E., 2011).

5. COMPOSITION MONITORING

Once the composition process is in execution, the composition monitoring stage allows specifying and reasoning about the monitoring properties during process execution. The composition process is specified using the EC and is then used to instantiate, verify and execute the composition process, Figure 14-1. The instantiation phase involves finding a solution to the composition process using the EC reasoner and the instantiated plan is then executed using the execution engine, Figure 14-2.

The proposed monitoring framework, Figure 14-3, works during the composition process execution and is divided into three phases, as discussed below:

5.1 Properties Specification.

The specification phase requires the user to specify the functional and non-functional properties that need to be monitored to identify anomalies or needs for KPI’s measurement. The properties that need to be monitored are added to process description either at the process design (if they are already known, Figure 14-1) or they can be added to the process specification at the execution time. In the later case, the process specification is updated and an updated instantiated solution is sought, in order to verify any conflicts and to get an updated execution plan as a result of process change during execution, Figure 14-3→1.

Properties that can be monitored include the functional aspects such as monitoring the invocation and execution order or they can be based on non functional aspects such as temporal aspects requiring to monitor the response time for a service, delay between successive invocations of the service or monitoring invocation time-frame for a service. Furthermore, the properties can also be based on data such as monitoring the data availability, validity and expiry or based on the security properties such as monitoring the data integrity, confidentiality, access-control etc. The choice of highly-expressive EC formalism even allows to combine the properties related to temporal, security and other aspects such as monitoring the data validity and access control within specific time frame which may be needed for instance, during dynamic task delegation (see (Gaaloul et al., 2010) for details). We briefly present EC models for some of these properties in Figure 15.

\[
\text{Happens}(\text{StartInvoke}(S1), \text{time1}) \land \text{Happens}(\text{EndInvoke}(S1), \text{time2}) \land \text{time2} - \text{time1} = \text{SomeTimeValue} \rightarrow \text{Happens}(\text{Terminateprocess}(), \text{time2}).
\]

\[
\text{Happens}(\text{InvalidateResponse}(\text{service}), \text{time}) \rightarrow \text{Happens}(\text{SendAlertNotification}(), \text{time}).
\]

Figure 15: Monitoring properties specification

The first axiom in the model shown in Figure 15 specifies a monitoring property for monitoring the response time for a service. If the difference in the occurrence of the Start and End event is greater than SomeTimeValue, the process is terminated. The monitoring properties have the general form Property → Response and we will later discuss different response actions. The second axiom above specifies to send Alert notification once the response message from any service does not remain valid.

5.2 Detection and Effects Calculation.

Detection. The detection of the violations can be handled at different levels using the proposed framework. At a basic level, we first consider the violations to the execution plan, which is handled by maintaining an event repository, which
keeps track of all the messages exchanged between the composition process and the participating services during process execution. This repository is then used to find any mismatch between the temporal ordering of actual events and the ones mentioned in the initial instantiated plan. Using the basic detection technique, it is possible to find violations to the execution plan or the invocation and execution order of the services. However, such a detection level may not be useful in detecting data values based or other low-level violations, as using the EC the process is modeled at an abstract level. This can be handled by also abstracting the processing of verifying the data values and other low-level service details by using EC fluents. The detection phase may thus require the execution engine support (for instance checking data validity, Figure 14-2).

Then, in order to detect the monitoring properties added at the execution time (e.g. based on external events not there in the initial instantiated plan), the “abduction reasoning” mode can be used by adding the newly added events and monitoring properties to the process model and re-invoking the reasoner. In case of no conflict and violation, the reasoner returns an updated plan based on the added events and monitoring axioms. However, if there is some conflict based on addition of new events or if the newly added monitoring property is not satisfied, the reasoner returns a set of unsatisfied clauses highlighting the error. The detection phase may thus also require the reasoner support, Figure 14-3→1.

**Effects Calculation.** Once a violation to some monitoring property is detected, the effects calculation phase is responsible for calculating the side-effects this violation has on the overall process flow. This allows to prioritize the violations based on their impact and it may be possible to ignore some violations, for instance if the response time delay for a service has no effect on the overall process goal and other functional and non-functional properties associated with the composition process. Adding the partial plan with the violation to the initial plan and re-invoking the reasoner achieve the effects calculation. Although, the process may seem similar to the detection of monitoring properties added at the execution-time, there is one major difference; instead of using the “abduction reasoning” we use “deduction reasoning” in the effects calculation phase. This may further allow foreseeing any anomalies, which may not be evident now but may happen later in the process execution. The effects calculation phase thus requires the support from the EC reasoner to perform deductive reasoning, Figure 14-3→1.

### 5.3 Response.

The response for the monitoring properties may involve some domain specific actions to cater for or measure the KPI’s and other parameters (such as logging, performance evaluation etc) needed for the successful process execution. Then, in order to cater for the monitoring violations detected at the execution time, different recovery actions can be used in-order to recover from the violation. These actions may include to ignore the violation, terminate the process, re-invoke or substitute the service, find an alternative solution based on current process state or backtrack to some previous state and then seek an alternative solution etc. Below, we briefly discuss the alternative-path as a recovery action as it highlights the need for a reasoning-based approach.

The recovery process is handled by first adding the current process state (with the violation). The reasoner is then re-invoked to perform abductive reasoning for the goal. However, it is not always possible to recover from a violation and respecting the associated constraints and composition goal. As a result, some constraints may require to be relaxed and the proposed approach allows to identify the conflicting clauses and hard-constraint if a recovery solution is not possible. The proposed approach thus preserves all the functional and nonfunctional constraints associated with the composition process (unless needed to be relaxed) while performing recovery. Further, the proposed approach allows both to find a new solution based on the current process state (thus specifying what steps should be taken now to recover from the violation and hence termed forward recovery) or to backtrack to some previous activity (if possible) and try to find a new from there. The response phase may require the execution run-time support (for instance actions such as logging, KPI’s measurement etc, Figure 14-3→2) and may also require the support from the DECReasoner in order to do abductive reasoning for actions such as finding alternatives, Figure 14-3→1.

### 6. Implementation Details

The EC models for the proposed framework are specified using the discrete EC language (Mueller, 2006) and all the models mentioned earlier can be directly used for reasoning purposes.

#### 6.1 The ECWS Application.

In order to abstract the EC models from the process-designer and automate the composition process specification, verification and monitoring, we have implemented a Java-based application, called ECWS, that provides a user friendly interface for specifying the composition design. It allows specifying different components and control/data flow between them.

Further, the ECWS application allows to generate EC models for the specification and will automatically invoke the DECReasoner to reason about the generated EC models. The resulting models returned by the DECReasoner are then displayed to the user both in the RAW form and by parsing them and aligning them with a time-modeling approach, Figure 16. The proposed ECWS application for the process specification is still in early phases and only serves as a proof of concept prototype and does not handle process execution and monitoring.
Enhancements to DECReasoner.

One important limitation of DECReasoner is the time taken for EC to SAT encoding which exponentially increases with the increase in time-points and introducing complex axioms involving multiple free variables, as we discussed in (Zahoor et al., 2010a; Zahoor, Perrin, & Godart, 2010b). In this work, we have thus modified the encoding process by two approaches. First, the process encoding is done only once during the instantiation phase of the DISC framework and encoding for any subsequent changes to the process description, such as during process execution or during effects calculation phase of the proposed monitoring framework, is added to the initial process encoding.

Further, we have thoroughly analyzed and modified the C language code for the encoding process to improve performance. Profiling the encoding process, we identified that the hashing function (DictHash) is not that efficient as it tries to calculate the hash-value based on first 6 characters of the input symbols. However, the structure of input symbols (in general) is such that only last few characters differ from other symbols. This results in a lot of collisions/chaining and subsequent use of strcmp takes all the time. By just changing the hash function to calculate the hash based on last 6 characters of the input symbol, we can avoid a lot of hashing conflicts and this improves performance. The updated hash function is shown below and the changes can be downloaded from DECReasoner official Website.

```
int DictHash(Dict *d, char *symbol)
{
    unsigned char s[6]; size_t len; len=(size_t)strlen(symbol);
    memset(s,0,6);
    if (len > 6) {
        memcpy(s,symbol+(len-6),6);
    } else {
        memcpy(s,symbol,(size_t)len);
    }
    return (int)(((s[1]+s[5]+(s[0]+s[4])*(long)256) +
                   (s[3]+s[2]*(long)256)*(long)481) % d->size);
}
```

6.3 Performance Evaluation Results.

In this section, we will detail the performance evaluation. The tests were conducted on a MacBook Pro Core 2 Duo 2.53 Ghz and 4GB RAM running Mac OS-X 10.6. The DEC reasoner version 1.0 and the SAT reasoner, relsat-2.0 were used for reasoning. For the performance evaluation results, we consider two case studies. First, we consider a composition process being setup to semi-automate the disaster plan for the Australian National Herbarium (ANH), Canberra, as discussed in (Zahoor E., 2011). We further complicate the example by increasing the number of components (and conditions) and adding the same control/data flow constructs and temporal and security constraints for the newly added components (and conditions).

The time taken for process instantiation (solution computation using relsat solver) and the effect of proposed modifications to the encoding process is shown in Figure 17, with X-axis representing the number of components and the Y-axis representing the time in seconds. The performance results indicate that the modified encoding process (based on the changes we proposed) result in significant performance improvement. The solution computation using the relsat is very efficient.

A detailed discussion about performance evaluation of zchaff solver for the process verification and the effectiveness of the proposed unsatisfiable-core filtering approach can be found in (Zahoor et al., 2012).

7. CONCLUSIONS

In this paper, we have presented the DISC framework, which provides a constraint-based declarative approach for self-healing Web services composition The proposed DISC framework has three main stages; Composition design, Instantiation and execution, and Composition monitoring.

http://hipore.com/ijsc
The composition process starts when the user specifies the composition design using a user-friendly interface allowing him to drag and drop components and provide constraints. The composition design is back-end by an EC event calculus model having direct mapping for all the components and specified constraints. The specified composition design is then used during the instantiation and execution phase to either identify any conflicts (such as deadlocks) or to provide a set of solutions (plans) for the composition process. If the design is conflict-free and a set of solutions is returned during the instantiation, a particular plan (user selected) is then executed and is monitored during the composition monitoring phase. The proposed approach is declarative and advocates the need for a unified formalism to handle different process stages. As a result, it allows recovery actions such as re-planning to find alternatives in case of monitored run-time violations and allows the process to be self-healing. We have also presented implementation details and performance evaluation results.

8. REFERENCES


http://hipore.com/ijsc


Flexible pattern monitoring for WS-BPEL through stateful aspect extension. In IEEE International Conference on Web Services, ICWS'08, pp. 577-584.


Authors

Ehtesham Zahoor did his PhD from Université de Lorraine, France in 2011. He is currently working as an Assistant Professor at National University of Computer and Emerging Sciences, Islamabad, Pakistan. He has been an active researcher in the domain of Web services composition and author of numerous peer reviewed conference/journal publications.

Kashif Munir did his PhD from University of Innsbruck, Austria in 2009. He was a post doctoral researcher at Telecom Bretagne, France from February 2011 to December 2012. He is an Assistant Professor at National University of Computer and Emerging Sciences, Islamabad, Pakistan. He is a reviewer of Elsevier Computer Networks and International Journal of Computer Mathematics and author of numerous peer-reviewed conference and journal publications.

Claude Godart is a Professor at University of Lorraine, France. He has lead and made researches about business process management and Web services composition in complex settings: flexible process models, transactional processes, Web services composition and governance with QoS properties, trusted execution of processes in clouds. He is author and co-author of more than 200 papers, and adviser of around 30 Ph.D. theses. He contributed to several software developments, including the first version of Bonita (http://www.bonitasoft.com). He is member of the editorial board of several journals (e.g. IEEE TSC, IJSC) and member of many conference program committees (e.g. BPM, ICWS, Cloud, SCC).

Olivier Perrin is full professor at University of Lorraine, France. He has expertise in Business Process Modeling, Web services compositions, workflow management, and enterprise application & data integration. He has published several articles in international journals, and conferences and he was involved in many international and European projects. He has been member and/or chair of many international conferences, and reviewer for IEEE journals.
A DIGITAL SECURITY CERTIFICATE FRAMEWORK FOR SERVICES

Samuel Paul Kaluvuri¹, Hristo Koshutanski², Francesco Di Cerbo¹, Renato Menicocci³, Antonio Maña²
¹SAP Labs France, ²University of Malaga, Spain, ³Fondazione Ugo Bordoni, Italy
samuel.paul.kaluvuri@sap.com, hristo@lcc.uma.es, francesco.di.cerbo@sap.com, rmenicocci@fub.it, amg@lcc.uma.es

Abstract
Service Oriented Architectures have facilitated a paradigm shift in software provisioning models: software gets consumed as a service, providing enormous benefits to both service providers and consumers. However, a major barrier for a wider adoption of the new service provisioning model in business and security-critical domains is the lack of security assurance of a service. Although current security certification schemes typically provide the required assurance, applying them to service environments is practically infeasible, a major reason being the natural language representation of the security certificates which becomes a major obstacle for adoption in typical service environment scenarios such as service discovery and service composition. To overcome the limitations of existing security certificates, we present a full-fledged framework for the realization of the concept of digital security certificates for services. The framework comprises a language for machine processable digital security certificates (DSC), and the concept of a security profile that allows the generation of uniform DSCs and a tool that aids in the generation of the DSCs.

Keywords: Security Assurance, Digital Security Certificates, Security Certification, Web Service Security

1. INTRODUCTION
Service Oriented Architectures have facilitated a profound change in the manner software is provided and consumed. Software, now, is offered as a service relieving consumers from the complexity of procuring and maintaining large scale IT infrastructures and at the same time facilitating inter-organizational interoperability (Gartner, 2012). However, in such provisioning models, the consumer ceases to have any control over the software provisioned or its operational environment. This lack of control, especially in critical domains such as finance, defence and healthcare, raises concerns about the security of these services (Gartner, 2012). Traditionally, consumers were able to gain the required assurance by establishing Service-Level Agreements (SLAs) with the service providers. But service landscape is dynamic and in such environments SLAs do not scale well. More often than not, the consumers end up agreeing to the terms and conditions of the service provider rather than ascertain whether their requirements - both functional and security - are met by the service.

In traditional software provisioning models, security certification of software by trusted third party entities is used to provide security assurance to consumers. Certification schemes such as Common Criteria (The Common Criteria Recognition Agreement, 2009) are well established and quite successful in providing the required security assurance to consumers in a scalable manner. However, current certification schemes result in certificates that are represented in natural language, which do not cope well with the dynamic service environment.

Service consumers should be able to compare the (certified) security features of a service with their security requirements and in addition to compare the (certified) security features of service offerings from different service providers. However, current security certification processes result in certificates that are represented in natural language and filled with legalese; this prevents any sort of automated reasoning to be performed on them.

In order to bring security certification to the service environment in a manner that they can play a constructive role in the service selection and service consumption process, several modifications to the current state of the art are necessary (Lotz, Kaluvuri, Di Cerbo, & Sabetta, 2012). Among them, we focus on the aspects related to the security certificate generation and representation. In this regard, we have identified the following key requirements from a service consumer perspective that need to be addressed in order to facilitate security certificate adoption in service landscape.

Requirement 1: The security certificates must be machine processable in order to allow automated reasoning to be performed on them.

Requirement 2: Security certificates should contain enough information about the certified entity so that they can cater to consumers with varying levels of security knowledge, such as regular users with limited security understanding to security experts of organizations. In other words, it is necessary that the certificates are descriptive (Wallnau, 2004), meaning, that they describe with sufficient...
details the security features of their services, together with supporting evidences.

Requirement 3: Mechanisms must exist in order to bind a service and its security certificate, given that a service implementation can change, while maintaining the same external interface or API. Consumers would need to have trustworthy and dynamic means to verify whether a service implementation they are using is the certified one.

The main objective of a security certificate of a service is to provide security assurance to potential consumers. The existing security certification schemes certify products at varying levels of assurance. The number of levels and the type of assurance depends on the certification scheme. Each certification scheme fixes these levels based on carefully designed and selected criteria. The levels of assurance provided by the current schemes distinguish certified products based on security features (such as FIPS-140) or the rigour of evaluation (such as CC, CPA) (The Common Criteria Recognition Agreement, 2009). However, security assurance is a multi-dimensional property that amalgamates assurance gained from security features of the product, evaluation of the product, etc. Hence, the levels of assurance provided by the existing schemes are a part of the overall security assurance that a consumer gains from the certificate. We have identified some of the aspects that can impact the security assurance provided to a consumer: i) the rigour of the evaluation; ii) the trust that the consumer has on the certificate issuer and evaluators; iii) the extent of information that is provided in the security certificate (trust through transparency). Hence, the security certificate representation should cope with different certification schemes with varying notions of “assurance”. This is yet another requirement that needs to be addressed by the security certificate.

Requirement 4: A security certificate should be certification scheme independent, so that different certification schemes could generate security certificates that are represented in a standard manner.

We present the concept of a digital security certificate (henceforth referred as CRT), that in our view addresses the mentioned security certification issues in service environment. In addition we propose the concept of a digital security certificate Profile (DSC Profile) that allows production of certificates that satisfy certain semantic and syntactical requirements.

The paper is structured as follows: Section 2 describes a service environment scenario for the application of the CRT concept, while Section 3 presents the state of the art in security certification and digital certificates. Sections 4 and 5 depict different aspects of the CRT, respectively its conceptual model and its technical representation. Section 6 explains the issues that arise from the CRT representation model and Section 7 presents the concept of a CRT Profile that overcomes the issues of the CRT representation model. In Section 8 we present the tool that can be used to generate CRTs by the Certification Authorities and in Section 9 we discuss the advantages of using the CRT concept and the impact it can have on the current security certification practice. Section 10 concludes the paper.

2. USECASE PRESENTATION

In order to illustrate the expressivity of the CRT, we present a scenario of a cloud storage service, “TitaniumBox”. We model the TitaniumBox service on the popular cloud storage service “Dropbox” (Dropbox inc, 2013), however, we do make certain assumptions where necessary and these are stated clearly. We assume that the TitaniumBox service has undergone a security certification process similar to a Common Criteria certification. In the following sections, we describe the service architecture and the security properties that the service claims to possess.

2.1 Service Description of TitaniumBox

The service allows consumers to store their files remotely. In addition, it also performs versioning of the files that are uploaded by the consumers. The service exposes an interface through a WSDL file that enables consumers to invoke the service. The WSDL file provides information only regarding the interface, as in the data types and the operations available to a consumer. The service is actually composed of multiple components that work together to provide the storage service to the end-user. A more detailed description of the system is shown in Figure 1. As can be noted from the figure, the service uses another storage service, Amazon S3, provided by an external party (Amazon). Let us assume the service runs on the Apache CXF framework on a server that runs on Debian operating system platform, in addition, the service uses a relational database, MySQL to store user specific information.

When such a service undergoes security certification process, similar to the Common Criteria, it results in a security certificate that is captured in human readable form. Clearly, such a certificate does not allow automated reasoning to be performed, thus not supporting assessments and comparisons among alternatives which is essential in scenarios such as service discovery.

2.2 Security Properties of TitaniumBox

The TitaniumBox service provider claims to have implemented several security mechanisms to provide certain

http://hipore.com/ijsc

Figure 1. TitaniumBox Architecture Overview
security properties to consumers. The service provider claims to have implemented the AES-256 encryption to provide the confidentiality of the user data when stored. This mechanism is implemented within the component “FileValidation” component. However, these claims alone will not provide any assurance unless the certification authority evaluates the service and certifies that the service indeed has the security properties claimed by the providers. The certification authorities must represent these certified security properties at varying levels of abstraction to cater to consumers with varying levels of security knowledge, needs and requirements.

3. STATE OF THE ART

A survey of the current security certification schemes reveals that there are quite a few established and successful schemes such as Common Criteria for Information Security (CC) (The Common Criteria Recognition Agreement, 2009), Commercial Product Assurance (CPA) (National Audit Office, 2013) and so on. Security certification schemes can be broadly classified based on the domains that they are applicable in, the recognition of the certification schemes, the descriptive or normative character of the issued certificates and so on (Wallnau, 2004).

Table 1 presents an overview of the existing security certification schemes. It can be observed that several certification schemes are specific to a particular domain such as Mobile Security, Network Security and so on, while a few schemes such as Common Criteria (CC), Commercial Product Assurance (CPA), and First Level Security Certification (CSPN) are applicable across the different domains. Among these broadly applicable schemes, the CC is a widely recognized (Common Criteria, 2012a), used (Common Criteria, 2012b), multi-domain (Turner, 2009), partially descriptive certification scheme (Beckert, Bruns, & Grebing, 2010). Schemes such as CPA, CSPN have different criteria towards the certification but they result in certificates that are represented similar to the CC Scheme. And hence we examine the Common Criteria scheme further for our analysis.

3.1 Common Criteria Certification Scheme

The CC scheme can trace its origin from schemes such as Trusted Computer System Evaluation Criteria (TCSEC) (USA Department of Defense Standard, 2013) also referred as “Orange Book”, the Canadian Trusted Computer Evaluation Criteria and Information Technology Security Evaluation Criteria. It has unified its predecessors with a standard set of criteria for security evaluation. It is designed in a very generic way which enables the CC scheme to certify products that range from software, firmware to hardware.

It decouples the specification of security functional and assurance requirements, this is in direct contrast with the approach of its predecessor TCSEC where the security functional and assurance requirements are coupled (USA Department of Defense Standard, 2013) together to provide a “balanced” assurance regarding a system. This decoupling is needed as CC targets a commercial security market as opposed to the “Orange Book” which was limited to products designed to be used by the US Government organizations. The CC scheme is very generic, as it aims to evaluate security of products that range from software, firmware to hardware. It avoids an all or nothing benchmark, by providing security assurance at varying levels, called Evaluation Assurance Levels (EAL), this provides flexibility for product vendors to get their product certified at lower assurance levels and improve the EAL over time.

The CC scheme is primarily “claims” based as it allows product vendors to describe the security functional requirements (SFRs) that are met by the product and to prove that the set of SFRs are able to counter the threats identified for a Target of Evaluation. This information is captured in a document called “Security Target” (Common Criteria, 2012b) (CC-ST) which is often seen as the descriptive part of the CC certification (Beckert et al., 2010). The product vendor then specifies the set of Security Assurance Requirements (SARs) in order to provide a certain Evaluation Assurance Level.

The CC scheme also enables consumers or consumer groups to specify their security functional and security assurance requirements for a certain type of product in an implementation independent way. This information is captured in a document called “Protection Profile” (CC-PP). Product vendors can then design products that conform to a certain protection profile.

The standardized SFRs and SARs are the “common” part of the CC scheme allowing, in theory, comparison of security features of certified products. However, in practice, the comparison of products which have different “claims” can be very hard. This is due to the representation of the CC related documents (CC-PP, CC-ST) in natural language, which is often filled with legalese and heavy security jargon making it rather complex to understand for non-security experts. Hence it becomes extremely difficult to determine if a particular product satisfies a consumer’s security requirements and to compare different products against their requirements. In (Lotz et al., 2012), it is observed that natural language representation of certificates is not a scalable solution when we consider service marketplaces such as Google Apps for business, Salesforce etc. There, the analysis of thousands of application/service offerings and their human readable security certificates would represent an unsustainable burden for customers; though the availability of security certificates could represent a means to gain assurance on offerings, it cannot facilitate any sort of automated reasoning such as compare and/or contrast the security properties of different services. It also prevents...
consumers to search for services based not only on their functional but also security requirements.

Although there are few “digital security seals” such as the TRUSTe privacy seal (Benassi, 1999), McAfee SECURE seal (McAfee, 2007) and so on, these seals are purely normative statements regarding the security feature of an entity, which can be seen as a step towards digital security certificates, but cannot provide any meaningful assurance to consumers as they do not contain any information regarding the certified entity.

In order to overcome these limitations we have developed a language to represent a security certificate in a structured, machine processable format. The conceptual model on which the language is based on, is presented in the following section.

4. CONCEPTUAL MODEL OF DIGITAL SECURITY CERTIFICATE

The CRT, which is the conceptual model for a Digital Security Certificate (DSC), is designed to capture information emanating from security certification processes. In particular, we have considered the CC scheme, as it is the most widely used scheme currently. The CC-ST, which is the descriptive part of the CC scheme, serves as a foundation for our CRT. However, we have extended this significantly, in order to make it machine processable and suitable for service-specific needs. In contrast to CC, and other existing certification schemes, the digital security certificate is designed to be completely descriptive (Wallnau, 2004), and hence it contains the description of the certified entity, the security properties of the certified entity, the evaluation specific description and User defined extensions.

Definition 1: The digital security certificate is a tuple $\text{CRT} = (\text{SD}, \text{SPS}, \text{ESD}, \text{UDE})$ where, $\text{SD}$ is the service description, $\text{SPS}$ is the security property specification, $\text{ESD}$ is the evaluation specific description and $\text{UDE}$ is User defined extensions.

4.1 Service Description

In CC-ST, a unique link cannot be made between the security properties and the assets that they secure, since, e.g., the assets (a native concept in CC) are described in natural language and without identifiers, and the security properties (not a native concept in CC) have to be derived by analysing the whole security problem definition. In order to overcome this, we adopt an asset-centric approach with explicit references between the assets and the different elements in the security certificate.

Definition 2: An Asset, $a$, is an entity that is of some value to the consumer or the provider. Assets can be data, applications, the IT equipment on which the service operates or even users of the Information System.

The CC-ST contains the Target of Evaluation (TOE) that describes the system that is being certified and the boundaries of the evaluation are indicated, albeit in an ad-hoc manner. However for a machine readable certificate there should be a clear distinction between the system that is
being certified and the aspects of the system that are subject to evaluation. It is of utmost importance in service based systems, due to the fact that services can be easily composed of external services and this information should be a part of the service description but clearly marked as outside the scope of evaluation.

The TOE in a CC-ST also contains the system architecture, the different components that compose the system among other information such as configuration in which the system is evaluated, the underlying IT architecture etc., and this is represented in natural language accompanied by architecture diagrams. This poses another issue in representing the TOE in a machine processable manner. In order to address these two issues, we introduced an element called Target of Certification (TOC) that describes the service being certified, while the TOE describes the part of the Target of Certification that is evaluated.

**Definition 3:** A Target of Certification is a tuple \( \text{TOC} = \langle \text{ACI}, \text{DM}, \text{TT} \rangle \) where \( \text{ACI} \) is the Asset-Component Identification, \( \text{DM} \) is the Deployment and Implementation Model and \( \text{TT} \) is the TOC Type.

**Definition 4:** An Asset-Component Identification is a tuple \( \langle A, C, \alpha \rangle \), where \( A \) is the set of all the assets identified for the TOC, \( C \) is the set of all the components in the TOC and \( \alpha \subseteq A \times 2^C \) maps each Asset with a set of Components.

**Definition 5:** The TOE is a subset of the Asset-Component Identification. \( \text{TOE} \subseteq \text{ACI} \)

The TOC Components are an integral part of the TOC as they allow TOC to be expressed in a modular and structured manner. A TOC Component comprises an abstract model of the Component, the Component Model: it can be as simple as just containing the interfaces of the component, or a more detailed specification of the internal dynamics of the component as deemed sufficient by the Certification Authorities. It must also contain technical specifications of the Component, again at the level of abstraction as deemed sufficient.

**Definition 6:** A Component is a tuple \( C = \langle \text{CM}, \text{TM} \rangle \), where \( \text{CM} \) is the component model and \( \text{TM} \) is the technical model.

We do not provide a formal definition for \( \text{CM} \) and \( \text{TM} \) as such concepts can be considered atomic for the sake of our work.

A Security Problem Definition \( (\text{spd}) \) is essential in a security certificate as it provides the rationale for securing the assets. The rationale for securing the assets can stem from the threats that are identified for the assets by the service provider or from the service provider’s security policy (which in turn could be due to compliance to regulations etc.).

**Definition 7:** The security problem definition is a tuple \( \text{spd} = \langle \hat{A}, \text{spr} \rangle \), where \( \hat{A} \subseteq A \) is a set of assets that need to be secured and \( \text{spr} \) is a security problem rationale for securing the assets.

**Definition 8:** The security problem rationale \( (\text{spr}) \) is a union of threats \( T \) and service provider’s security policy \( \text{SSP} \) \( \text{spr} = T \cup \text{SSP} \).

The service description must contain the description of the certified system, the part of the system that is evaluated and the rationale for protecting the assets that are identified.

**Definition 9:** The Service Description is a tuple \( \text{SD} = \langle \text{TOC}, \text{TOE}, \text{SPD} \rangle \) where, \( \text{SPD} \) is the set of security problem definitions \( (\text{spd}) \).

### 4.2 Security Property Specification

The CC-ST contains a vast amount of information but is often presented in heavy jargon; this rarely allows a consumer (a non-security expert) to get a high level perspective of the security features provided by the software/service. Hence we introduced a new element in the CRT model called as “security property specification” which enables a fine grained description of the security property that originates from a multi-layered model. It comprises of different elements, from abstract security properties to concrete security mechanisms.

**Definition 10:** An Abstract Security Property \( \hat{p} \) is an atomic security attribute for an asset.

For example, abstract security properties can be confidentiality, integrity, availability, authenticity, non-repudiation, utility, privacy and so on. Since abstract security properties by themselves do not convey any information on how the property is applied, there is a need for contextual information. Hence we define Contextual Security Property.

**Definition 11:** A Contextual Security Property is an abstract security property realized in a certain context. \( \hat{p}_c = \langle \hat{p}, c \rangle \), where \( c \) is a context.

Contexts depend on the abstract security property. Abstract security properties that are data centric such as the CIA triad can have contexts such as transit, rest and usage (such as Confidentiality in rest and Integrity in transit). However these properties still lack a subject, i.e., no indication of “what” is being secured. This is addressed by the certified security property.

**Definition 12:** A certified security property \( p \), is a contextual security property \( (\hat{p}_c) \) applied on a set of Assets \( (\hat{A}) \), \( p = \hat{p}_c \times \hat{A} \).

The (certified) security property provides a high level overview of how an asset is secured. But this does not provide any information on how the \( \text{SPD} \) are addressed. This is overcome by using the concept of “Security Objectives” similar to the CC scheme. A security objective, \( so \), counters, mitigates or detects a \( \text{spd} \) that is identified for the \( \text{TOE} \) and contributes to the realization of a security property \( p \) for the \( \text{TOE} \).
5. Representation of a Digital Security Certificate

In order to realize the conceptual model of the digital certificate CRT, we have developed an XML-based language that enables the representation of the certificate in a machine processable form, which from henceforth we refer to as an ASSERT. A detailed description of the latest version of the schema can be found in (Koshutanski et al., 2013) and in this section we will explain its most relevant elements using the example introduced in Section 2.

5.1 SAML as Container of ASSERT

The management and exchange of the ASSERTS is an important consideration for a successful implementation of a certification ecosystem, i.e., production, maintenance, consumption of certificates. In this context, the container of the ASSERTS assumes significant importance as it is needed to encapsulate the certificate data into an interoperable format that can be used with existing web service standards and technologies. We have chosen the SAML standard (SAML Specification, 2012) as a container because it is widely used in decentralized systems for its support for request and exchange of SAML Assertions, either for authentication or authorization of entities, or any attributes of an entity. The SAML standard has support for several standard profiles for usage of SAML tokens in specifications such as WS-Security (OASIS, 2006), WS-SecurityPolicy (OASIS, 2007a), WS-Trust (OASIS, 2007b), etc. These aspects make SAML a good choice to be a container for exchanging ASSERTS in service environment. We use the SAML Assertion tokens to encapsulate ASSERT-specific data.

Figure 2 shows the main elements of the SAML assertion token structure where the Statement element defines an abstract statement of an assertion. Similar to how SAML authentication and authorization decision statements extend the abstract Statement element, we extended this element to provide a statement about a service’s description, its security property along with the corresponding evidence. The standard field Issuer in the SAML token is used as a means to capture the ASSERT Issuer’s identity (the certification authority issuing the ASSERT). The Subject field represents the identity of the certificate requester, which in most cases will be the service provider. And the validity conditions and the signature data are inherent to all security tokens.

5.2 ASSERT Structure

Figure 2 shows main elements of the ASSERT structure. It has three major elements: ASSERTCore, ASSERTType-Specific and UserDefinedExtensions. The ASSERTCore part contains elements that are independent of the evaluation of a service, i.e. the SD and the SPS elements. The evaluation information in the conceptual model, i.e. ESD, is contained

---

http://hipore.com/ijsc
in the ASSERTTypeSpecific element, while the UDE is captured in its namesake element UserDefinedExtensions.

The ASSERTCore element contains, in addition to the SD and SPS, elements such as ServiceBinding that provides a robust link between the service and its ASSERT, CertificationProcess that provides information related to the certification process of a given service, and a textual description of the certificate in the ASSERT4Humans element where the certified service and the certified property are explained in natural language for end-user comprehension.

The AssertSigner element identifies the entity that signs the ASSERT, while the PerformedBy element in the CertificationProcess identifies the entity who performed the service evaluation. Since multiple entities can be involved in a certification process, for example the ASSERT issuing process and service evaluation process may be undertaken by different entities, we provide this feasibility so as to increase the accountability during the production of certificates. In order to better illustrate the ASSERT language we provide code excerpts from the ASSERT of the example we provided in Section 2.

5.2.1 Service Description in ASSERT Core

The SD in the conceptual model is mapped to the CertificationProcess element in the ASSERT language. It contains the elements such as TargetOfCertification and SecurityProblemDefinition which map to the TOC and SPD respectively in the conceptual model. In addition, we have incorporated an element called CertificationCriteria used to represent any specific criteria followed during the service certification process (e.g., compliance to regulations).

The TargetOfCertification element is depicted in Figure 3. The elements in the ACI are represented directly in the TargetOfCertification element i.e., the Assets, TOCComponents. It also contains the Type, DeploymentAndImplementationModel and Description providing textual description of the TargetOfCertification for end-user comprehension. We enforce the explicit identification of both the Assets and TOCComponents by making the use of the ID element mandatory. The set that maps assets with components, α, in the ACI is realized within the asset definition by mapping each asset to specific components using the TOCComponentRef (which is of type IDRef) to provide a binding between the assets and components.

The TOE is not represented as an explicit part of the service description in the ASSERTCore, but we use the flag InTargetOfEvaluation in the TOCComponent element that indicates whether the component is a part of the TOE, and avoids a duplicate representation of the components in both the TOE and TOC to have an optimized ASSERT.

5.2.2 Security Property Specification in ASSERT Core

The SecurityProperty element maps to the p element in the conceptual model. However, on the representation (language) level we have defined a single property certified
in ASSERT. Such “separation” of certified properties allow
us to have practical implications on management of
ASSERTs throughout their life-cycle, such as generation,
consumption (reasoning), and revocation of ASSERTs. For
example, if an ASSERT certifies two properties, say
“confidentiality in transit” and “confidentiality in storage”,
and during the ASSERT lifetime the given service does not
anymore comply/provide “confidentiality in transit” due to
some technical reasons, the certification authority has to
revoke the ASSERT although the second property may still
hold.

The SecurityProblemDefinition element in the ASSERT-
Core contains a list of ProblemDefinition, as shown in
Figure 4. Each ProblemDefinition is mapped to the spd in
the conceptual model. However, instead of a set of assets, the
ProblemDefinition contains a list of references to the
Assets defined in the TargetOfCertification.

Figure 5 shows the SecurityProperty element structure
consisting of an abstract security property realized in a
context and on a set of assets. The SecurityProperty
contains a NameID that defines a name identifier of the
described property. The NameID allows reference to
external ontologies to describe the certified security
property. The PropertyAbstractCategory defines the
abstract category of the security property. The
PropertyContext element defines a context in which the
abstract security property is realized. The Assets defines a
set of Asset elements on which the security property applies.
Each Asset element is a reference to an Asset definition in
the TargetOfCertification section. The SecurityObjectives
defines a set of SecurityObjective elements of the security
property. The main elements of the SecurityObjective are: a)
an identifier of the described security objective; b) a set of
SecurityProblemDefinitionRef each referring to a
ProblemDefinition defined in the SecurityProblemDefinition
section; c) Name that contains the name of the security
objective; d) Description which describes the security
objective. It is an implicit assumption that all
SecurityObjectives together contribute to the realization of

6. SEMANTIC COMPLEXITY AND
INCONSISTENCIES DUE TO THE
EXPRESSIVITY

Since the ASSERT language is designed in a way that it
is security certification scheme agnostic, there are major
issues that can arise out of that design choice:

- Facilitate comparison among security certificates.
  Given the flexibility and richness of certificate
  languages and ability to express similar security
  assertions in different ways, a certification authority
  may wish to define a certificate profile (e.g., by
  defining various certificate structure and content
  mandatory) to enforce uniformity of content of
  certificates when issued by accredited entities.
- Facilitate production of security certificates compliant
  with specific certification criteria. Given that a
  certificate language can support various certification
  schemes, a certification authority has to define its

http://hipore.com/ijsc
certification criteria in a certificate profile, so that all issued security certificates will conform to the criteria defined by the certificate profile.

- **Enable consumers to specify their security requirements for the services.** Similarly to CC-PP, the consumers or consumer groups may wish to define a certificate profile with domain-specific security requirements (criteria). When services conform to such certificate profiles, it eases the decision making process for the consumers as the conformance to a profile implies that their requirements are met by the service.

7. **Digital Security Certificate Profile**

A **DSC profile** is a mechanism to specify the contents and semantics of a class of DSCs. The main goal of a DSC profile is to provide suitable means for creation of certificates by ensuring semantic uniformity of certificates for a specific (domain of) certification capturing any certification and evaluation specific aspects, vocabulary of products certified, security properties, or other aspects relevant to the semantics of DSCs.

7.1 **Conceptual Model of DSC Profile**

In the following, we will introduce the profile structure (Montenegro, Maña, & Koshutanski, 2013). A DSC profile is composed of three parts: (i) **Certificate Template**: specification of the common structure and the values of specific fields mandatory for a given certificate class, (ii) **Semantic Rules**: specification of the semantics of the certificate class in the form of semantic rules, and (iii) **Vocabulary**: specification of vocabulary terms (ideally ontology-referenced terms) providing restrictions on use of vocabulary for language artefacts of security certificates of the given certificate class.

Figure 6 shows the abstract structure of the DSC profile. The three profile components provide certificates content uniformity in three different dimensions: certificate template ensures structural uniformity; semantic rules ensure integrity of intended semantics of certification; while certificate vocabulary ensures common ontology-based ground of terms and ranges of possible values of certification (in a given domain).

The certificate template is a partially filled certificate that establishes the common structure and content of all certificates created based on a certificate profile. Therefore, any certificate conforming to a DSC profile must include the fields, structure and values defined in the template of the profile. A certificate template specifies an incomplete realization of a structure with respect to a given certificate syntax (e.g., XML schema). It is used as baseline for creating new certificates.

Alternatively, a certificate template can be considered as a set of implicit (semantic) rules. These rules are simple and easy to understand. For this reason, it is not required to represent a template as a set of rules, but used as a certificate template – a more intuitive notion for expressing predefined structure and values of certificate elements.

The Semantic Rules define semantic constraints and dependencies between content of certificate artefacts within a given class of DSCs. While the implicit rules defined by the certificate template are enough for structure-wise restrictions (requiring an optional element be mandatory, constraining specific structure or content of certificate artefacts, etc.), there are cases where more complex restrictions are needed such as to express artefact dependencies or artefact content constraints.

Semantic rules represent a solution allowing to formulate rules to ensure integrity of an intended semantics of a given certificate class, i.e., preserving specific semantics of certification artefacts. Semantic rules can be formulated in rule-based languages (such as Schematron (Schematron, 2006) or variants of OCL (Object Constraint Language, 2012)) or imperative languages (such as Java or JavaScript) in function of the underlying certificate language and supported implementation. The choice of a language for expressing semantic rules has an important implication to achieve machine processability and reasoning of the rules. The language should allow rich fine-grained expression of patterns over certificates' content and structure.

The certificate vocabulary part of the profile provides a means to define and restrict use of vocabularies on different certificate artefacts. One of the goals of the vocabulary part is to facilitate integration of the certificate language with ontology terms coming from different domains of knowledge. The ontology integration will enhance the semantic robustness among all certificates conforming to a given profile, which have been diminished by flexibility and openness of security certificate languages (models). Ontologies provide not only a suitable source of semantically defined terms but also provide means to define relations between terms, and equivalences between different terms. That gives us a powerful way to query ontologies for different aspects of certification and related semantics. The certificate vocabulary section enables the use of static or dynamic vocabularies. A static vocabulary defines actual terms inside a profile. It is suitable for offline processing, but could be out-dated by an ontology evolution/update. In contrast, a dynamic vocabulary defines actual terms by means of a query over ontology, which requires Internet connection for online processing.

Figure 6. DSC Profile Structure
7.2 Representation of DSC Profile

We have realized a DSC profile structure tailored to the ASSERT representation, which we will refer to as ASSERT profile henceforth. We refer to (Koshutanski et al., 2013) for more details on the actual profile schema and in this section we provide the main aspects of a DSC Profile.

An example of an ASSERT profile structure is shown in Figure 7. The profile defines the following class of ASSERTS. The ASSERTTemplate defines that all ASSERTS conforming to this profile must: (i) Be for software-as-a-service (SaaS) model services, i.e., all ASSERTS must have TargetOfCertification element with an attribute Type qualified as “http://assert4soa.eu/ontology/-a4s-language#Software-as-a-service”; (ii) Be issued by the University of Malaga as issuer, i.e., all ASSERTS must have an ASSERTIssuer element with the defined value structure (conforming to X.509 subject structure) “O=University of Malaga,OU=Computer Science Department,C=ES”; (iii) Be produced by a test-based certification process, i.e. must contain ASSERT-E type-specific structure, but without defining any particular content for the structure. This means that ASSERTS stating conformance to the profile can contain any specific ASSERT-E content. The SemanticRules section defines one Schematron rule, which forces the security property abstract category value as defined in the SecurityProperty element in the ASSERTCore match the value of the PropertyName of Property definition of ASSERT-E.

Such an integrity constraint is difficult to enforce without a semantic rule. The ASSERTVocabulary defines two vocabularies – one for the PropertyAbstractCategory attribute of the SecurityProperty element, and another one for the PropertyContext attribute of the same SecurityProperty element. The first vocabulary defines static values of the CIA triad – Confidentiality, Integrity and Availability – as terms from an ontology specific definition, and marks those as mandatory. The second vocabulary defines optional values for the artefact PropertyContext, such as InStorage, InTransit and InUsage, as terms from an ontology-specific definition.

8. DSC MANAGEMENT TOOL

We will describe the necessary tool support for the core profile-based certificate management operations: profile-based creation of DSCs, and profile conformance verification of DSCs. These are the most relevant DSC management operations a DSC issuer would need to perform when issuing DSCs.

8.1 Profile-based Generation of DSC

Figure 8 shows the profile-based creation process. When a profile is selected and loaded, there is a pre-processing step for all dynamic vocabulary specifications. If some dynamic vocabulary specifications depend on other artefacts and values in order to be processed, these vocabularies are processed at the time when the issuer creates the corresponding artefacts.

Step 1: Initialize DSC content. Once the profile is processed, first duplicate the certificate template and create an initial certificate instance with an initial structure and content of the duplicated template data.

Step 2: Edit DSC content. Next step is the actual process of editing the certificate artefacts and creation of new artefacts as needed by the issuer. This step heavily relies on the use of certificate vocabulary defined in the profile. When an artefact’s vocabulary is specified as mandatory, the tool will enforce the choice of the vocabulary terms. Otherwise, if optional, the tool will recommend, suggest a choice of terms but leaving the issuer to specify own terms when he finds necessary.

Step 3: Profile Conformance Verification. The final certificate instance is verified for conformance to the profile (presented in the next subsection). All non-properly used artefacts and corresponding vocabularies will be reported. Step 3 will give a feedback to redo step 2 of the creation process. Repeat step 3 until the certificate instance conforms to the profile.
8.2 Profile Conformance Verification of DSC

A prerequisite to conformance verification is to ensure if the certificate instance conforms to the syntax of a given certificate model, that is, if the certificate instance is a syntactically valid certificate. Otherwise, the verifier should not proceed with the verification process. Figure 9 shows the three main steps of conformance verification process.

**Step 1:** Structure validation. DSC structure is validated if it contains all required elements and values as declared in the profile template.

**Step 2:** Vocabulary Validation. DSC vocabulary is validated for compliance with the vocabulary defined in the profile. Prerequisite to this step is to first process all dynamic vocabularies. That is, retrieving all certificate artefacts’ vocabulary terms from the corresponding ontologies by executing the queries. Once dynamic vocabularies are instantiated, all certificate artefacts’ vocabulary terms within the vocabulary part are checked against the corresponding artefacts’ content in the certificate instance. All certificate artefacts defined to have an optional (non-mandatory) vocabulary will not be verified for conformance.

**Step 3:** Semantic Validation. DSC structure and vocabulary is validated for compliance with the profile rules, that is, if all constraints are satisfied. All semantic rules are processed, checked if satisfied by the certificate structure and content. Since the semantic rules of the profile may depend on the actual content (vocabulary) of a certificate artefacts in order to determine the semantic integrity of the certificate content, it is important to verify vocabulary conformance first, and then the semantic rules conformance.

8.3 DSC Management Tool Realization

A DSC management tool has been developed to support the management of ASSERTS from the perspective of ASSERT issuers. The tool is called ASSERT Management Tool* (AMT). The AMT is designed to provide an intuitive and formative GUI for standard ASSERT management operations including complete support of multi-profile based ASSERT management. We will present the GUI of the AMT for the case of profile-based creation of ASSERTs.

Figure 10 shows the AMT designer view. The designer view provides content-centric ASSERT management: abstracting issuers from underlying XML representation. All mandatory ASSERT language elements are coloured in red for convenience of issuers. A help icon is shown to each element name describing the rationale of the selected element. The AMT has the ASSERT language schema built-in. A designer pane shows all language elements as buttons. Upon pressing a button, the corresponding data element is created. As we can see in Figure 10, the ASSERT profile has been already loaded and the corresponding profile vocabulary processed showing the vocabulary terms defined for the PropertyContext element of the SecurityProperty (ref. Figure 7). We note that in case of multiple profiles, the AMT loads each profile in a separate profile view (tab) so that the issuer can at any moment check if any of the profiles is well-formed and what messages are shown of profile validation.

9. IMPACT ON CURRENT SECURITY CERTIFICATION PRACTICES

Current security certification schemes are a result of harmonizing the security assurance requirements by security experts, government regulatory bodies and they provide a broad consensus on the assessment of software security in a manner that is both repeatable and observable. Each certification scheme has at its own criteria, chosen carefully so that it can cater to evaluating products that range from software, firmware to hardware. However, a detailed analysis of the existing criteria must be performed to verify if the current criteria is capable of providing assurance for services or it needs to be adapted. This analysis is outside the scope of this paper, while here we focus on the security certification practice. The current security certification practices would be profoundly impacted by the concept we proposed. The model allows certification results to be captured in a machine processable form thereby allowing automated reasoning to be performed on them. Automated reasoning on security certificates opens up hitherto unexplored scenarios such as instantaneous comparison of certified products, requirements compliance by certified products, fine grained and precise description of secured

---

* Available at http://proteus.lcc.uma.es/projects/assert4soa/software/

http://hipore.com/ijsc
elements within a certified product leading to less ambiguity or errors in interpretation of the security certificates.

In addition, the AMT tool can be used by the certification authorities to issue security certificates in a much faster manner reducing the previously time consuming process of producing certificate artefacts. In fact, the AMT tool and its output artefacts were evaluated by a focus group composed of domain experts, two Common Criteria certifiers and two experts in security certification. The activities comprised an explanation of the basic underpinnings for DSC, ASSERT and ASSERT Profile concepts, as well as an evaluation of the tool operations and outputs. The evaluation consisted of a questionnaire composed mostly by closed-answer questions using 5-points Likert scale, as well as a number of free answers questions. The latter allowed for the expression of feedback on specific DSC aspects, which was used later on for improving the AMT tool. The questionnaire was structured in two sections: the first part was devoted to usability assessment (Rubin & Chisnell, 2008), while the second aimed at assessing the suitability of AMT tool outputs and procedures (e.g. ASSERTs and ASSERT validation against an ASSERT Profile) for their application in a security certification process operations.

An analysis of focus group results can be summarized as follows. Regarding the usability of the AMT tool, it was assessed positively, especially considering its nature of research prototype. Nevertheless, several suggestions were proposed, and some of them have been incorporated in the prototype; others were deemed interesting for future commercial exploitation of the concept. With respect to the suitability of ASSERTs and AMT tool operations to security certification operations (especially considering Common Criteria), we will focus our attention on a selection of the focus group questions, which are:

- Q-15: From the point of view of a Certifier, an ASSERT is suitable to represent a typical security certificate.
- Q-16: The AMT speeds up the ASSERT management process.
- Q-17: The AMT improves the control over the ASSERT management process.
- Q-18: From the point of view of a Certifier, the automation of a typical security certificate management process is a priority.

Their proposed answers (and their mapping to the 5-points Likert scale) were: I fully agree (2), I mostly agree (1), I neither agree nor disagree (0), I mostly disagree” (-1) and I completely disagree” (-2). The answers are represented in Figure 11.

The answers to Q-15, Q-16 and Q-17 are particularly encouraging: Q-15 essentially confirms the suitability of the ASSERT and DSC concepts for representing security certificate contents, while Q-16 and Q-17 assess positively the AMT tool operations.

From Q-18 and open answer questions, it is possible to derive the following findings.

From the point of view of the technical quality of the AMT, most of the respondents generally agreed that the representation capability of an ASSERT is suitable from the certifier’s point of view. However, from the point of view of its adoption and relevance, the main output of the validation session indicates that current real world certification processes (and especially Common Criteria) are probably not ready to embrace digital certificates in their current forms. For Common Criteria, this is mainly due to its complexity and difficulty to automate its operations. However, the advantages of using ASSERT and DSC Tool can represent a stimulus for the evolution of a debate inside the Common Criteria community.

10. CONCLUSION

To conclude, we claim that the adoption can represent
significant benefits for an uptake of third-party service offerings in business-critical domains such as financial, defence and healthcare (Di Cerbo et al., 2012). The digital security certificates provide security assurance of services that would allay the security concerns of potential customers, which is one of the most relevant obstacles nowadays (Gartner, 2012). In addition, we presented the concept of a DSCP profile that helps in the generation of uniform CRTs. Finally, we presented a tool that helps in the generation of the CRTs by a Certificate Issuer.

11. ACKNOWLEDGMENTS
This work was partially supported by the EU-funded projects ASSERT4SOA (grant no. 257361), CUMULUS (grant no. 318580) and ANIKETOS (grant no. 257930) of Framework Programme VII of the European Commission. Portion of this paper appeared under the title "Security assurance of services through digital security certificates" in Proceedings of the 20th IEEE International Conference on Web Services (ICWS-2013), Santa Clara Marriott, CA, USA, June 2013, pp. 539—546.

12. REFERENCES


International Journal of Services Computing (ISSN 2330-4472)  Vol. 1, No. 1, October-December 2013


Authors

Samuel Paul Kaluvuri is a Research Associate in SAP Research at Sophia Antipolis, France. He is also pursuing a PhD from Telecom Paritech, Paris. Samuel received his Masters in Business Studies in Electronic Commerce from National University of Ireland, Galway. He has participated in SAP’s contribution to FP7 European projects PrimeLife, Avantssar, Assert4SOA and Aniketos.

Dr. Hristo Koshutanski received M.Sc. in Mathematics from Plovdiv University ”Paisii Hilendarski” in 2001 and Ph.D. in Information and Communication Technology from the University of Trento in 2005. He had a postdoctoral position in the field of security and trust management at CREATE-NET international research centre for 2005-2006. He holds EU Marie Curie EIF postdoctoral fellowship with a host institution the University of Malaga for 2007-2009. He has participated in several European projects in both Sixth Framework Program (FP6) and Seventh Framework Program (FP7) of the European Commission. He was a member of advisory board/interest group of the European project ONE-Open Negotiation Environment (FP6), and is an external ethics advisor of the European project SUPERHUB (FP7). His research interests span across: (i) Models of authorization, access control and trust management for business-based systems, business processes and dynamic coalitions; (ii) Security architectures for distributed, federated and cloud-based systems; (iii) Security assurance, certification and digital security certificates of cloud and Web services; (iv) Identity management in federated systems, (v) Public Key Infrastructures and Privilege Management Infrastructures (PKI/PMI). He has co-authored a number of scientific papers.

Dr. Francesco Di Cerbo received his PhD in 2008 from University of Genoa, Italy. In 2011, he joined SAP Research, in the Security & Trust practice. Previously, he was assistant professor at the Faculty of Computer Science at Free University of Bolzano-Bozen. He is author of a number of contributions to international ACM and IEEE conferences; he also served as Programme Committee member of conferences, and as reviewer for a number of journals. He was research associate for the Italian National Research Council. His current interests include mobile security, trust management and usage control mechanisms in SOA and cloud environments.

Renato Menicocci graduated with a summa cum laude degree in Electronic Engineering at the University of Rome “La Sapienza” in July 1991 with a thesis on Cryptanalysis. From 1992 to 2000, he was a Researcher of the Cryptology Group of Fondazione Ugo Bordoni (FUB). From 1996 to 1999, he was the Italian delegate at ISO/JTC1 SC27 WG2. From 2000 to 2002, he was a Cryptography and Computer Security Expert of the Rome Crypto Design R&D Centre of Gemplus. From 2003 to 2008, he was a FUB consultant for Digital Television Security and ITSEC/Common Criteria Security Evaluation. Since August 2004, he’s been cooperating with the Department of Electronic Engineering, University of Rome “La Sapienza”, where, from 2004 to 2006, he was under a research fellowship connected to SCARD (EU-FP6). Since 2008 he is a researcher of FUB, where his main interests are ICT Security Evaluation and Certification, especially under Common Criteria. He has been involved in ASSERT4SOA (EU-FP7) and is currently involved in CUMULUS (EU-FP7). He has been involved as a teacher in several academic contexts. He is author or co-author of several scientific works published in international journals.

Prof. Antonio Maña received his MSc and PhD degrees in Computer Engineering from the University of Malaga in 1994 and 2003, respectively. In 1995 he joined the Department of Computer Science of the University of Malaga where he is currently Professor in the Computer Science Department. His current research activities include security and software engineering, information and network security, application of smart cards to digital contents commerce, software protection and DRM. He has more than 100 peer-reviewed publications. He has participated in several EU funded projects and is currently the Principal Investigator of the UMA team in FP7 PARIS and CUMULUS projects, and has previously been involved in the FP7 ASSERT4SOA, SECFUTUR and OKKAM, FP6 SERENITY, GST and iAccess, and FP5 CASENET projects. Prof. Maña is member of the Editorial Board of the Intl. Journal of Electronic Security and Digital Forensics (IJESDF) and reviewer for several other international journals. He is member of different professional and scientific societies and workgroups and is actively involved in the organisation of research and educational activities.

http://hipore.com/ijsc
FORMAL METHODS FOR THE SPECIFICATION AND TESTING OF DATA-CENTRIC WEB SERVICES: A CASE STUDY

Iman Saleh*, Gregory Kulczycki†, M. Brian Blake*, Yi Wei†
*University of Miami, †Virginia Tech, ‡Microsoft
iman@miami.edu, gregwk@vt.edu, m.brian.blake@miami.edu, dawe@microsoft.com

Abstract
Web services allow organizations to capture their human and software-based capabilities as modular software components that are called remotely over a network. In such service-oriented settings, it is important to establish an agreement that sets the obligations of the service provider and the expectations of the service consumer. Since traditional approaches such as Service Level Agreements (SLAs) are loosely defined with respect to data integrity aspects, in previous work, we proposed a formal model for specifying data-centric Web services. The goal is to formally and unambiguously specify the service behavior in terms of its underlying data model and data interactions. In this paper, we present a case study where we use our model to specify and verify data-centric Web services. We demonstrate our proposed methodology using three state-of-the-art specification languages: JML, Dafny and RESOLVE. We also explore the use of our specification to automatically generate test cases for data-centric services. Our goal is to study the feasibility of our proposed model and also to pinpoint the challenges and limitations of current specification and verification tools.

Keywords: Formal methods; Web Services; Verification

1. INTRODUCTION
Unlike software components operating within an enterprise, the Web services model establishes a loosely coupled relationship between a service producer and a service consumer. Service consumers have little control over services that they employ within their applications. A service is hosted on its provider’s server and is invoked remotely by a consumer over the Web. In such settings, it is important to establish a contract between the service provider and the service consumer. The contract establishes a set of obligations and expectations.

Applying design-by-contract principles in the context of Web services addresses many practical challenges in current Web service development, and facilitates future development as formal verification tools mature. From the perspective of a service producer, a Web service must be designed and implemented to be broadly applicable. Quoting (Kaye 2003), “[The concept of broad applicability] suggests that the eventual uses of any given service can’t be predicted at the time the service is created... In other words: Design and implement your Web service interfaces to handle anything that might be thrown at them”. Accordingly, the service implementation must cover all possible invocation scenarios. One way to achieve this is by adding defensive checks to the service implementation. While these checks may ensure that the service is invoked with the correct input, they can affect the code efficiency in a negative way. As described in (Leavens and Cheon 2006), consider a binary search method that requires its array argument to be sorted. Checking that an array is sorted requires time linear with respect to the length of the array, but the binary search routine is supposed to execute in a logarithmic time. Design-by-contract techniques help avoid these inefficient checks by explicitly adding pre-conditions on the service inputs. It is then the consumer’s responsibility to satisfy these pre-conditions in order to guarantee that the service behaves as expected. Moreover, if these pre-conditions are formally defined, then the consumer can use automatic reasoning tools to check his/her service composition and to guarantee that it satisfies the pre-conditions. Furthermore, these checks are accomplished statically. That is, they are made without having to run the implementation that is based on the composition of services.

While the inclusion of preconditions is a best practice that is supported using current development technologies, the principle of formally specified contracts offer a much greater benefit in the future as tools for verification continue to evolve. Formally specified preconditions, along with postconditions and invariants can potentially be used to completely describe the behavior of a service. At a minimum, these descriptions provide web service consumers with a precise and unambiguous way to understand the service. But as tools mature, they also provide service producers with a way to statically verify the behavior of their services, significantly mitigating the need to perform run-time testing of their services. A few state-of-the-art tools are beginning to emerge that partially support these ambitious verification goals.

In previous work (Saleh et al. 2009a)(Saleh et al. 2009b)(Saleh et al. 2013), we proposed a data modeling and contracting framework for data-centric Web services. Our framework formally specifies a service with respect to its interaction with the data. We also provided pseudo-code of
the model implementation. In this paper, we show an implementation of our model and a case study using three of the state-of-the-art specification languages, namely JML\(^1\), Dafny (Leino 2010) and RESOLVE (Ogden et al. 1994). We choose these languages as they represent recent effort for developing specification languages and there’s an ongoing work to improve their verifiers. They also represent interesting examples of different specification approaches that we compare and contrast; JML is an extension of a programming language, Java. Dafny combines both procedural and functional programming and, RESOLVE has a sound mathematical foundation and enables the inspection of verification conditions, as we will be explaining later in detail. Our goal is to show the feasibility of our model implementation and to study the current challenges and limitations of formal languages and verification tools. Our experiment described here show that, despite the limitations of the tools, we could still reason about a service-based implementation and obtain some proofs of correctness. We also show how we can leverage the specifications in order to automatically generate test cases for data-centric services.

2. RELATED WORK

Semantic approaches have gained a lot of attention in Web Services community as a way to specify service capabilities. The W3C OWL-S\(^2\) standard for semantically describing Web services is an example of such an approach. Semantic techniques are based on description logic, which supports the definition of concepts, their properties and relationships. The reasoning tasks supported by description logic include instance checking, relation checking and subsumption (Baader et al. 2007). This makes techniques based on description logic suitable for solving problems related to the automatic discovery and composition of services, since these problems require matching between a semantically-annotated user query and a semantically-specified Web service. In contrast, our work is based on formal methods, which support verification of correctness of a computer program. For Web services, formal methods are suitable for solving problems related to compositional correctness and verifying that a service complies with its advertised specification. From a software engineering perspective, the semantic techniques and formal methods techniques are complimentary, as they address software validation and verification problems, respectively. While a semantic-based approach can validate that a service or a composition of services match a user query, a formal method approach can verify that the service or composition of services is implemented correctly with respect to that user query.

A related work in (Vaculin et al. 2008) identifies the need to make databases visible to service discovery and composition algorithms. The authors propose an extension to OWL-S standards to support the specification of data-providing services. A data-providing service is a read-only service that provides access to one or more possibly distributed and heterogeneous databases. The service’s main functionality is to retrieve data from a data store based on an input query. Their approach facilitates the automatic discovery of services by applying ontological reasoning. They suggest describing data sources as RDF views over a shared mediated schema. Specifically, the local schema of each data source is mapped to concepts of a shared OWL ontology, and its terms are used to define RDF views describing the data sources in the mediated schema (Vaculin et al. 2008). Their approach is useful in matching a service with a user query based on ontology-based reasoning. As stated before, they use semantic techniques and hence their approach do not address correctness issues or reasoning about the service side-effects. Another related work presented in (Deutsch and Vianu 2008) handles the specification of interactive Web applications and focuses on specifying Web pages and user actions. The proposed data model incorporates temporal constructs to specify browsing paths between pages and application behavior in response to user actions such as clicking a button or browsing through hyperlinks. This approach is useful in verifying properties like page reachability and the occurrence of some events. This approach is working from a process perceptive and an input-boundedness restriction is assumed to guarantee that the verification operation can be done in polynomial time. In our work, we are specifying and reasoning about so-called stateless Web services by modeling the state of the overall system, which necessarily includes the data store. We model the underlying data store as a sophisticated global variable, so that we can reason about how it is modified between a service request and response. We do not attempt to specify all states of a data store, as we focus on correctness and verification of data-related side-effects after a service call.

The Tisa language recently proposed in (Rajan et al. 2009) employs a similar methodology by applying formal methods techniques to specify temporal service policies. Policies include non-functional regulations and privacy rules that should be maintained by a service implementation. The language supports reasoning about correctness of a service composition with respect to these policies. In our work, our goal is to reason about correctness with respect to the functional data behavior of a composition of service. In general, our approach enhances the advertised interface of a service by including the specification of its data behavior.


3. A CONTRACTING FRAMEWORK FOR DATA-CENTRIC WEB SERVICES

We propose a framework to support modeling and contracting of data-centric Web services. Our framework enables contract-based composition of services as shown in Fig. 1. Based on our framework, the service developer abstracts the data model and annotates the service with a contract that specifies the data requirements and the service’s side effects. The contract is machine-readable and hence automatic reasoning techniques can be used at this stage to ensure the correctness of the service implementation with respect to the contract. Assisted with tools, the developer encodes and publishes this contract along with the service’s WSDL3 file. At service invocation, a consumer consults the contract to understand the service behavior, ensure correct service invocation, and interpret the service result and effect. Again, automatic reasoning can be used at this stage to ensure the correctness of a service-based composition. Reasoning on the consumer side guarantees that correct inputs are formulated and hence avoiding service requests that will return errors or cause unintentional side effects.

![Figure 1. Data Modeling and Contracting Framework for Data-Centric Web Services](image)

By applying design-by-contract techniques, our framework enables the service consumer to ignore the data and service implementation details. It also allows the service providers flexibility in creating or modifying implementations for their services. This separation between specification and implementation promotes software modularity and consequently promotes service reuse.

4. A FORMAL DATA MODEL

We model a data source as a set of entities, where each entity is a set of records. In addition to a unique record identifier (key), a record can have zero or more attributes. This model leverages current data models that we surveyed including the relational and object-oriented modeling of databases, and some earlier efforts for formally specifying databases (Souto 1994)(Fisher 2007).

```plaintext
class GenericDataModel
    attribute entity1: Set(GenericRecord1)
    attribute entity2: Set(GenericRecord2)
    ...  
    attribute entityn: Set(GenericRecordn)

operation GenericRecordi
    findRecordByKey(key: GenericKeyi)
    requires (GenericKeyi is the key for GenericRecordi)
    ensures (result.key = key and result in this.entityi)
    or result = NIL

operation Set(GenericRecord1)
    findRecordByCriteria(values1: Set(Ti1), values2: Set(Ti2), ... , valuesn: Set(Tin))
    requires (Tij is the type of the jth attribute of GenericRecordi)
    ensures \forall rec in result, rec.attrj in valuesj
    and result in this.entityi

operation GenericDataModel createRecord(gr:GenericRecordi)
    ensures result.entityi = this.entityi U gr

operation GenericDataModel deleteRecord(key: GenericKeyi)
    ensures result.entityi = this.entityi - this.findRecordByKey(key)

operation GenericDataModel updateRecord(gr:GenericRecordi)
    requires this.findRecordByKey(gr.key) ≠ NIL
    ensures result.entityi = this.deleteRecord(key).createRecord(gr)
end GenericDataModel
```

Listing 1. The Data Model Template

We adapt the CRUD (Create-Read-Update-Delete) (Kilov 1990) model to include functional descriptions of the basic data operations. The reader is referred to (Saleh et al. 2009a) and (Saleh et al. 2009b) for a detailed description of our model. The model is used to formally specify a service’s interactions with its underlying source(s). Listing 1 presents a data model class template that we provide to help developers in specifying a data source.

Throughout our work, we use Hoare-style specification (Hoare 1969) to define an operation’s pre/postconditions. A precondition is specified using a ‘requires’ clause and a postcondition is specified using an ‘ensures’ clause. The result keyword is used to denote an operation’s output. The ‘#’ prefix is used to denote the value of a variable before the invocation of the operation. We use these notations in Listing 1 to specify the data operations. These operations are defined as pure mathematical functions that have no side effects.
effect, which enables us to use them to specify the Web services.

5. THE PAYPAL EXPRESS CHECKOUT CASE STUDY

5.1 GENERAL DESCRIPTION

The PayPal online payment solutions expose a set of services to facilitate the integration of electronic payment within e-commerce websites. The integrations steps are described within a so-called Express Checkout flow depicted in Figure 2.

As shown in the figure, in the flow, the buyer:
1. Chooses Express Checkout by clicking the Check out with PayPal button
2. Logs into PayPal to authenticate his or her identity
3. Reviews the transaction on PayPal
4. Confirms the order and pays from your site
5. Receives an order confirmation

Behind the scene, this scenario is implemented by invoking three of PayPal Web services. Table 1 contains the description of these three services. We are using a simplified version of the flow as a composition case study that we formally model and specify using the proposed framework.

5.2 THE DATA MODEL

Given the PayPal description and documentation of these services, we infer the data model of the underlying database for the Express Checkout services as shown in Listing 2. The model is based on the template defined earlier in Listing 1.

As shown in Listing 2, we define the underlying Express Checkout record model, represented by the TransRecord class, as consisting of a token, a payment transaction amount transAmount and the corresponding payer information captured by the payerInfo attribute. A payment has a status represented by the paymentStatus attribute. The token attribute is a timestamped token that is used by the three Express Checkout services to relate different services calls to one payment transaction. It is unique and hence we choose to use it as the transaction record key. This example shows how our model can reuse Web service data types defined in WSDL files; in the above listing, for example, we are reusing PayerInfoType which is a complex data type used by different PayPal services to hold the payer information such as name, shipping address, email and others. This practice is very useful in minimizing the effort of modeling a service and ensuring that the model complies with the original service design.

---

5.3 Individual Service Contracts

The data model is used to annotate services with formal specifications represented as data contracts. Listing 3 shows how we use the PayPal model to expose a data contract for each individual service in the PayPal flow. Our specification is intended to be complete; i.e. any programming or state variable that is not explicitly specified in the service contract is assumed to be unchanged after the service execution.

In our specification of the PayPal services, we begin by defining a model variable ppdm of type PayPalDM representing the underlying data store. A model variable is a specification-only variable (Cheon et al. 2005) that is used in conjunction with programming variables to model the state of the system. We include the model variable in each service specification to reflect the fact that all services are reading and updating the same data store and hence service returns the payer information related to that transaction record. The service does not change the data model since the contract does not explicitly define a ‘modifies’ clause.

It should be noted here that our example uses services from the same service provider. However, our proposed approach can be used to specify a composition of heterogeneous services that interact with different sources of data. In that case, we simply need to define a data model variable for each data source used in the composition.

5.4 Specification of Service Composition

We use formal specification to annotate the composition of services with a global data contract. The contract represents this flow in pseudo-code, but we believe this representation is foundational to more specific flow languages such as BPEL 5 or OWL-S. We assume a global variable token of type string. The token is the transaction record, identified by the service input gToken, and this record should have a non-null payer information attribute. In other words, the service is called when the payer information has been captured and saved in the data model. The contract’s ensures clause specifies that the service returns the payer information related to that transaction record. The service does not change the data model since the contract does not explicitly define a ‘modifies’ clause.

As a demonstrative example, consider the contract of the getExpressCheckoutDetails service in Listing 3. The contract ‘requires’ clause specifies that, before the service invocation, the underlying data model should have a transaction record, identified by the service input gToken, and this record should have a non-null payer information attribute. In other words, the service is called when the payer information has been captured and saved in the data model. The contract’s ensures clause specifies that the service returns the payer information related to that transaction record. The service does not change the data model since the contract does not explicitly define a ‘modifies’ clause.

It should be noted here that our example uses services from the same service provider. However, our proposed approach can be used to specify a composition of heterogeneous services that interact with different sources of data. In that case, we simply need to define a data model variable for each data source used in the composition.

Listing 3. The Individual Data Contracts of the PayPal Services

The data model is used to annotate services with formal specifications represented as data contracts. Listing 3 shows how we use the PayPal model to expose a data contract for each individual service in the PayPal flow. Our specification is intended to be complete; i.e. any programming or state variable that is not explicitly specified in the service contract is assumed to be unchanged after the service execution.

In our specification of the PayPal services, we begin by defining a model variable ppdm of type PayPalDM representing the underlying data store. A model variable is a specification-only variable (Cheon et al. 2005) that is used in conjunction with programming variables to model the state of the system. We include the model variable in each service specification to reflect the fact that all services are reading and updating the same data store and hence capturing dependency and compositional effect of services on that data store. Consequently, the state in our case is represented by service inputs and outputs in addition to the data store model variable. To simplify the specification, we also define a model variable rec of type TransRecord that is used to specify a transaction record, whenever needed.

As a demonstrative example, consider the contract of the getExpressCheckoutDetails service in Listing 3. The contract ‘requires’ clause specifies that, before the service invocation, the underlying data model should have a transaction record, identified by the service input gToken, and this record should have a non-null payer information attribute. In other words, the service is called when the payer information has been captured and saved in the data model. The contract’s ensures clause specifies that the service returns the payer information related to that transaction record. The service does not change the data model since the contract does not explicitly define a ‘modifies’ clause.

It should be noted here that our example uses services from the same service provider. However, our proposed approach can be used to specify a composition of heterogeneous services that interact with different sources of data. In that case, we simply need to define a data model variable for each data source used in the composition.

Listing 4. The pseudocode and the formal specification of the composition of services for the PayPal Express Checkout flow

We use formal specification to annotate the composition of services with a global data contract. The contract describes the intended behavior, from an integrator’s point of view, for the flow of services based on the individual data contracts of each of the participating services. The flow implementation and the global contract are shown in Listing 4. Listing 4 represents this flow in pseudo-code, but we believe this representation is foundational to more specific flow languages such as BPEL 5 or OWL-S. We assume a global variable token of type string. The token is the
timestamped value, described before, that relates different service calls to the same transactions. The contract’s ‘ensures’ clauses specify that the checkout flow creates a new transaction record with payment amount equal to the input payment amount. Also, the flow result is equal to the payer information associated with the newly created record in case the transaction is processed. Otherwise, the result is \textit{Nil} and the payment transaction is marked as denied.

6. \textbf{IMPLEMENTATION DETAILS}

6.1 \textbf{REQUIREMENTS}

In order to implement the proposed model using a specification language, the language design must support the following constructs:

- Specification-only variables (a.k.a ghost variables): The data model in our framework is defined as a specification-only variable that is used to represent the underlying database and specifying the service-to-data interactions.

- Side-effect free methods: These are used to define the basic data operations supported by the data model. These operations are used in the specification of a service interface. Operations used in the specification must not change the state of the program.

- Specification of interface-only methods: Programmers depend on the APIs signatures in order to call services within their code. A specification language used to specify Web services must hence support a mechanism to specify interface-only methods. A verifier in this case can be used to ensure the consistency of the specification and not to prove its correctness.

In the following sections, we discuss parts of the implementation of the PayPal Express Checkout composition using the Dafny, JML and RESOLVE languages. The complete implementations can be found in (Saleh 2012). We have also made an attempt to implement the modules using Spec#, however, Spec# does not provide constructs for user-defined theory types and hence we could not define our data model.

6.2 \textbf{IMPLEMENTATION MODULES}

The specification and verification of the PayPal Express Checkout flow entails the implementation the three modules shown below in Figure 3:

- Module (1): The PayPal data model.
- Module (2): The data contract for each of the three PayPal services.
- Module (3): The implementation and the data contract of the PayPal Express Checkout that composes the three PayPal services.

In the following sections, we discuss parts of the implementation of these three modules using the Dafny, JML and RESOLVE languages.

6.3 \textbf{DAFNY}

Dafny (Leino 2010) is a class-based specification language. A Dafny class can declare variables, methods, and functions. The language supports specification-only variables through ghost variables. It also supports user-defined mathematical functions that can be used in writing specifications. The language has a verifier that translates it to the Boogie intermediate verification language (Barnett et al. 2006). A Boogie tool is then used to generate first-order verification conditions that are passed to the Z3 theorem prover (De Moura and Bjørner 2008). The types supported by Dafny are booleans, mathematical integers, references to instances of user-defined generic classes, sets, sequences, and user-defined algebraic datatypes. Specifications in Dafny include standard pre- and postconditions, framing constructs, and termination metrics.

Listing 5 is an excerpt of the model implementation in Dafny. The implementation includes a definition of the \texttt{TransRecord} class. Due to the abstraction level of the Dafny language, we have simplified the model in terms of the data types used; sequences of integers for example are used to represent strings. Sequences are also used to represent the collection of record representing the \texttt{transEntity} attribute in the model class. The \texttt{transEntity} attribute is defined as a ghost variable as it is declared for specification-only purposes. For the same reason, the methods supported by the model are defined using Dafny’s mathematical functions. In Dafny, mathematical functions are declarative, side-effect free functions that can be used to write specifications. The domain of a function is defined by a requires clause. The reads clause gives a frame for the function, saying which objects the function may depend on. The decreases clause gives a termination and the function’s body defines the value of the function (Leino 2010). We
hence implemented a body for each function in the data model to define its value.

```java
class TransRecord {
    var token: seq<int>; // used to represent a string
    var transAmount: int;
    var payerInfo: seq<int>;
    var paymentStatus: seq<int>;
}

class PayPalDM{
    // A 'sequence' is used to define collection of records
    ghost var transEntity: seq<TransRecord>;

    function isValid(): bool
    reads *;
    { (* <= [transEntity]) &&
      (forall j :: j >= 0 && j < |transEntity| ==
        transEntity[j] != null)
    }

    function findRecordByKey(key: seq<int>, i: int): TransRecord
    requires isValid();
    requires i >= 0 && i <= |transEntity|;
    decreases |transEntity|-i; //specifies that the recursion terminates
    {
      if i == |transEntity| then null
      else if transEntity[i].token == key then transEntity[i]
      else findRecordByKey(key, i+1)
    }
}
```

Listing 5. An excerpt of the data model implementation in Dafny

<table>
<thead>
<tr>
<th>Method/Function</th>
<th>Verification Time (sec)</th>
<th>Verification Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>isValid()</td>
<td>0.024</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>expressCheckoutFlow</td>
<td>0.172</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>setExpressCheckout</td>
<td>0.081</td>
<td>Postconditions unverifiable</td>
</tr>
<tr>
<td>getExpressCheckoutDetails</td>
<td>0.057</td>
<td>Postconditions unverifiable</td>
</tr>
<tr>
<td>doExpressCheckout</td>
<td>0.089</td>
<td>Postconditions unverifiable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method/Function</th>
<th>Verification Time (sec)</th>
<th>Verification Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>isValid()</td>
<td>0.012</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>findRecordByKey</td>
<td>0.022</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>findRecordByCriteria</td>
<td>0.030</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>findRecordIndex</td>
<td>0.020</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>deleteRecord</td>
<td>0.035</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>createRecord</td>
<td>0.012</td>
<td>Passed successfully</td>
</tr>
<tr>
<td>updateRecord</td>
<td>0.022</td>
<td>Passed successfully</td>
</tr>
</tbody>
</table>

Table 2. Verification Results using the Boogie Verifier

6.4 JML

The Java Modeling Language (JML) is a specification language that is used to specify Java modules. JML annotations are appended to Java code as comments proceeded by the at-sign (@). JML uses a ‘requires’ clause to specify a method’s pre-conditions and an ‘ensures’ clause to specify the post-conditions. The ‘result’ variable denotes the output of a method. The ‘old’ prefix denotes the value of a variable before the method invocation. In JML, side-effect free method labeled as pure can be used within a specification.

Listing 6 is the data model implementation in JML (we have omitted the definition of the TransRecord class to avoid repetition). An array is used to represent the collection of record representing the transEntity attribute in the model class. Similar to Dafny, the transEntity attribute is defined as a ghost variable as it is declared for specification-only purposes. The methods supported by the model are defined as abstract pure methods. In JML, side-effect free method labeled as pure can be used within a specification.
The authors of (Sitaraman et al. 2011) present the RESOLVE language and its verifying compiler. RESOLVE is used to write specified object-oriented code and provides tools for both generating the verification conditions and proving simple ones. The compiler has been recently implemented as a Web tool (Cook et al. 2011).

We give here a brief description of the language, its structure and tools. The Resolve Tutorial can be consulted for more detail7. The language has its built-in specifications that are written using universal mathematical notations. The RESOLVE compiler translates the code and the specification into Java code that can be compiled using Java compiler. The language is based on mathematical theories of programming data types such as integers, strings and booleans. These theories are used in writing the formal specifications and verifying them. In addition to the theories, RESOLVE supports different types of code units. A Concept in RESOLVE defines the mathematical model of a data structure. For example, a stack in RESOLVE is mathematically modeled as a sequence of strings. The stack Concept specifies operations such as pop, push and depth. An Enhancement is used to add custom functionalities. For example, an Enhancement can be used to add a stack reverse operation to the stack Concept. Both Concept and Enhancement units do not provide an implementation; they only provide the specifications. A Realization unit on the other hand provides the implementation of a Concept or an Enhancement. This organization of RESOLVE units enables decoupling the implementation from the specification.

We use a Concept to implement the PayPal data model, an Enhancement to specify the flow composition, and finally a Realization is used to provide the corresponding implementation. The RESOLVE Concept definition is shown in Listing 7. The complete code can be found in (Saleh 2012).

As shown in the code, RESOLVE defines a set of parameter modes. The ones we used include:

- preserves – the value of the incoming value will be preserved.
- replaces – value will be replaced by some other variable.
- updates – the value will be changed in an unspecified way.

To verify that the Express Checkout composition satisfies its specification, we first use the RESOLVE compiler to generate Verification Conditions (VCs). The VCs are a series of logical implications such that proving these implications is necessary and sufficient to demonstrate that the implementation is correct (Smith et al. 2009). The RESOLVE compiler generates the VCs in a user-friendly format that facilitates human inspection. They can also be
generated in a syntax accepted by the Isabelle proof assistant (Nipkow et al. 2002) or the RESOLVE integrated prover.

Given an assertion and an implementation, the RESOLVE verifier applies proof rules, replacing code with mathematical assertions and applying some simplifications. Assuming the soundness of the proof system, if the final assertion can be reduced to true, this implies that the first assertion is correct and hence the implementation satisfies the assertion. Since we used the RESOLVE Web tool, we were not able to collect accurate response time data as we did in the case of Dafny and JML. The tool is however relatively fast and the response time is in the order of seconds for program interpretation and VCs generation.

7. Analysis and Discussion
In this section, we discuss our experience using the three specification languages, we analyze the differences and similarities among them, and their suitability for specifying Web Services.

7.1 Language Constructs
First, to implement the data model, the language that we use must allow defining a specification-only variable representing the model. Both Dafny and JML provide ghost variables for this purpose. Ghost variables are theory-typed variables that are defined only for specification purposes. RESOLVE doesn’t have the explicit notion of global ghost variables; any variable in RESOLVE that is mathematically founded can be used in the specification.

To define and specify the data operations, the language must also support mathematical functions, or side-effect free methods, that can be used in the specifications. Dafny’s mathematical functions are used for that purpose. The body of a mathematical function in Dafny defines its postconditions. Consequently, we had to implement the data operations using Dafny’s functions instead of simply specifying them as originally intended. Once defined however, these implementations can be easily reused across different models. JML on the other hand conveniently provide spec files where interface-only methods can be defined and specified using JML assertions. RESOLVE provide a similar approach through the use of Concepts which also define interface-only methods. It’s worth noting here that we made some trials to implement our model using Spec#, however, the language is lacking the necessary constructs to define theory types and hence we could not use it to define the data model.

7.2 Verification Process
Out of the three languages, RESOLVE is the only one that provides a human-readable form of the Verification Conditions. This enables a programmer to inspect the VCs and detect any specification error or discover assertions that may be proven true but does not reflect the programmer’s intentions.

```
1 Concept PayPal_DM_Set; uses Std_Integer_Fac, Modified_String Theory, Std_Boolean_Fac, Set_Theory, Std_Char_Str_Fac;
2 Type Family PayPal_Record is modeled by Cart_Prod
3 Token: Char_Str;
4 Payment_Status: Integer;
5 Payer_Info: Char_Str;
6 Trans_Amount: Integer;
7 end;
8 end;
9 exemplar R;
10 constraint R.Token /= 0 and
11 (for all Ri: PayPal_Record, if Ri/= R then
12 Ri.Token /= R.Token) and
13 -- Payment_Status is modeled here using integers
14 -- 0 = 'denied', 1 = 'in-progress', 2 = 'processed'
15 (R.Payment_Status = 0 or R.Payment_Status = 1
16 or R.Payment_Status = 2) ;
17 initialization ensures R.Token /= empty_string;
18 Type Family PayPal_DB is modeled by
19 Powerset(PayPal_Record);
20 exemplar db;
21 constraints true;
22 initialization ensures true;
23 Var Rec: PayPal_Record;
24 Var PPDM: PayPal_DB;
25 operation Set_Express_Checkout(preserves
26 sPayment_Amount: Integer;
27 updates return:Char_Str);
28 updates PPDM, Rec;
29 ensures Rec.Token /= empty_string
30 and Rec is_in PPDM
31 and Rec.Payer_Info /= empty_string
32 and Rec.Trans_Amount = sPayment_Amount
33 and Rec.Payment_Status = 1
34 and return = Rec.Token
35 and PPDM = Singleton(Rec) union PPDM;
36 operation Get_Express_Checkout_Details(preserves
37 gToken: Char_Str;
38 replaces answer: Char_Str);
39 preserves PPDM, Rec;
40 requires Rec is_in PPDM and Rec.Token = gToken;
41 ensures answer = Rec.Payer_Info;
42 operation Do_Express_Checkout(preserves dToken: Char_Str;
43 preserves dPayment_Amount: Integer;
44 updates return:Boolean);
45 updates PPDM, Rec;
46 requires Rec is_in PPDM and Rec.Token = dToken;
47 ensures ((return = true and Rec.Payment_Status = 2 )
48 or (return = false and
49 Rec.Payment_Status = 0 )));
50 and Rec.Token = #Rec.Token
51 and Rec.Payer_Info = #Rec.Payer_Info
52 and Rec.Trans_Amount = #dPayment_Amount
53 and PPDM = #PPDM without
54 Singleton(#Rec) union Singleton(Rec);
```

Listing 7. A RESOLVE concept specifying the PayPal data model and Express Checkout operations

The VCs generation component of RESOLVE is both sound and complete (Sitaraman et al. 2011)(Wayne Heym 1995). The verification component of RESOLVE is still...
evolving and can currently be used to prove some of the simple VCs. The component is both sound and complete. On the other hand, the verification process of both JML and Dafny is neither sound nor complete due to limitations in the current implementations of the verifier. For example, we have intentionally introduced errors in the Dafny implementations and the verifier failed to detect these errors. Similarly, the verification process in JML failed to verify the implementation when a control flow statement is introduced as the verification space grows beyond the verifier capabilities.

7.3 TOOLS

Dafny uses the Boogie verifier, which is a command-line tool. It is also integrated with the Visual Studio IDE to provide a real-time checking of assertions. This enables detecting programming errors while coding. A Web interface is also available to try the language and its verifier on simple examples (Sitaraman et al. 2011). The language does not have yet a compiler and hence programs written in Dafny can only be verified but not executed.

To verify JML code, we use ESC/Java2 as it is relatively matured compared to other tools and has an active community. JML verification tools however have limited capabilities. They support a subset of the JML language and they don’t work with recent versions of Java, specifically with generics. Currently, there’s an effort to develop a new generation of tools, called OpenJML, which is based on OpenJDK and support recent versions of Java.

Finally, RESOLVE provides both a command-line tool and a Web interface compiler/verifier. The Web interface is particularly convenient to use and provide sample Concepts that can be adapted and reused. The RESOLVE compiler transforms the code into Java and can be used to generate the VCs for inspection. Using the Web tool, a realization can be marked with VCs, at approximate places, so that the user can connect the VCs with the code (Cook et al. 2011).

7.4 LEARNING CURVE

We share here our experience in using the three languages in terms of ease of use and learning the language constructs. The Dafny language is relatively easy to learn since it has very limited set of constructs. The language combines both procedural and functional programming and hence familiarity with both is needed in order to use the language effectively. Dafny supports a limited set of data types and hence it’s the programmer’s responsibility to compose these types into complex ones, when needed. JML is easy to learn for Java programmers as it uses similar syntax and is well integrated with the Java language. It defines a set of theory types that the programmer needs to learn.

RESOLVE defines its own syntax and program structure and hence its learning curve is steep relative to Dafny and JML. The language however defines mathematical model and related theories for simple programming types such as integers, floats, booleans and strings that can be easily reused. Many examples are also provided through the Web tool including the implementation and specification of basic data structures such arrays, stacks and queues. Table 4 summarizes the differences and similarities among the three languages.

Table 4. Comparison of the three specification languages

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dafny</th>
<th>JML</th>
<th>RESOLVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCs generation soundness</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>VCs generation completeness</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>VCs Inspection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Global Specification Variables</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Compiler</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ease of Learning</td>
<td>Easiest</td>
<td>Average</td>
<td>Hardest</td>
</tr>
<tr>
<td>Integration with a Programming Language</td>
<td>None</td>
<td>Java</td>
<td>None</td>
</tr>
</tbody>
</table>

8. AUTOMATIC TEST GENERATION

In the previous sections, we described how we used the specification of data-centric services in order to apply static analysis of code and prove code correctness. Formal methods research has also recognized the benefits of leveraging the formal specification in order to automatically generate test cases (Tretmans and Belinfante 1999)(Tillmann and De Halleux 2008).

Figure 5. Test case generation from formal specifications

For testing data-centric services, the generation of interesting database states becomes one of the main testing challenges. We propose using our formal model to generate
these states. The general proposed methodology is depicted in Figure 5. As shown in the figure, the service specifications are fed into a constraint solver that uses the specification to generate interesting test values for the code variables as well as the data. These test values are used to populate a test database before running the unit tests.

To evaluate the proposed methodology, we conducted an experiment where we study the effect of leveraging the data specification on the quality of tests generated by an automatic test generation tool. The experiment is conducted as follows:

1. A book rental application is implemented using C# Web services. We modeled the underlying database as a global variable, shared by all services, and we formally specified it using our proposed data model.
2. All services are specified using C# contracts. The contracts define preconditions, postconditions, invariants and the specification of non-null variables. These constructs are used to define how the services interact with the data model variable in terms of reading and/or modifying records.
3. We apply the Pex automatic test generation tool (Tillmann and De Halleux 2008) on the specified code and data model. The tool leverages the specification in order to automatically generate test cases.
4. We measure the code coverage achieved by the generated tests, at the following levels of specifications:
   - L0: No specification, this level acts as a baseline.
   - L1: Specifying only the non-null types.
   - L2: Adding both loops and class invariants to L1 specifications.
   - L3: Specifying preconditions in addition to L1 specifications.

As part of our exploratory experimentation with the tools, we applied our proposed methodology on two services that are part of the book rental application. Namely, the create_user service that creates a new administrator record given a username and password, and return_book, which marks a book as returned given the book’s ISBN. The results of testing these two services are depicted in Figure 6.

The figure shows the code coverage percentage measured at the different levels of sophistication for the specifications; L0 through L3. The results show the increase of code coverage as the specification is made more exhaustive. We actually reach 100% of code coverage for the create_user service at L2, i.e. with partial specification. While we couldn’t run the tool for a significant number of services due to the tools’ limitations, the result we obtained using the case studies are highly promising. By capturing the specification of the underlying data infrastructure, our proposed model enables the generation of interesting data states that exercise the different code branches and consequently achieves higher code coverage.

9. Conclusion and Future Work

In this paper, we present our effort in implementing a data model and contracting framework for Web services using state-of-the-art specification languages. With the current state of the formal specification languages and tools, it is still not possible to fully specify and verify every property of a large-scale service composition with respect to the requirements. It is however possible to verify some properties, as it is the case with our proposed model, where we focus on specifying and verifying the data aspect. Our experiment shows that we are able to verify some of the correctness properties using the current tools. There has been also a significant progress in this area. Some recent efforts aim at integrating specification techniques into current mainstream programming languages, the C# contracts are an example. While they still lack the necessary constructs for defining abstract data types and tools for verifying complex assertions, they are however useful in detecting some logical errors in the code and enhancing automatic testing (Fähndrich et al. 2010). An interesting extension of our analysis is to study the learning curve and utility of these languages when used by developers of Web service composition. We are also performing more experimentation on automatically generating test cases from the specifications and evaluating the quality of the generated tests in terms of the achieved code coverage and the ability to detect code defects.

10. References


Computer Systems and Software Engineering


Tretmans, J. and Belinfante, A. 1999. Automatic testing with formal methods.


Authors

Iman Saleh is an assistant scientist at the University of Miami. Her research interests include data modeling, Web services, formal methods, big data, and cryptography. Saleh has a PhD in software engineering from Virginia Tech. She’s a member of ACM, IEEE, and the Upsilon Pi Epsilon Honor Society for Computer Science at Virginia Tech. Contact her at iman@miami.edu.

Gregory Kulczycki is an adjunct professor of computer science at Virginia Tech. His research areas include component-based software development, software engineering, and formal methods. Kulczycki has a PhD in computer science from Clemson University. He’s a member of the IEEE Computer Society and the ACM. Contact him at gregwk@vt.edu.

M. Brian Blake is a professor of computer science and concurrent professor of electrical and computer engineering, and human genetics at the University of Miami. His research interests include service-oriented computing, workflow systems, and software engineering. Blake has a PhD in information and software engineering from George Mason University. He’s a senior member of IEEE and an ACM Distinguished Scientist. Contact him at m.brian.blake@miami.edu.
Yi Wei is a software engineer at Microsoft. His research interests include service-oriented computing, cloud computing, and agent-based modeling and simulation. He received his PhD in computer science from University of Notre Dame. He is a member of ACM and IEEE. Contact him at dawe@microsoft.com.
A LOCATION-AWARE SERVICE SELECTION MODEL

Ying Li, Zhiling Luo, Jianwei Yin
Zhejiang University, China

cnliving@zju.edu.cn, luozhiling@zju.edu.cn, zjuyjw@zju.edu.cn

Abstract
This paper introduces a service selection model with the service location considered. The location of a service represents its position in the network, which determines the transmission cost of calling this service in the composite service. The more concentrated the invoking services are, the less transmission time the composite service costs. On the other hand, the more and more popular big data processing services, which need to transfer mass data as input, make the effect much more obvious than ever before. Therefore, it is necessary to introduce service location as a basic feature in service selection. The definition and membership functions of service location are presented in this paper. After that, the optimal service selection problem is represented as an optimization problem under some reasonable assumptions. A shortest-path based algorithm is proposed to solve this optimization problem. At last, the case of railway detection is studied for better understanding of our model.

Keywords: service selection; service composition; service location; shortest path; big data

1. INTRODUCTION

In the Service Oriented Architecture (SOA), complex services can be easily composed using individual services from various service providers. Service selection, one of the critical problems of service composition, is generating considerable interests in recent years in several computer science communities. Service selection determines the non-functional qualities of composite service in varieties directions, such as cost, reliability and reputation. A large number of methods for service selection are presented by different researchers and communities. So there is a considerable amount of service selection approaches have been proposed by different researchers and communities. Some of these approaches have even already been used in practice.

However, the era of big data brings new challenges to service selection. The concept of big data comes from the research on Database, which represents that the size of data involved in computing is huge (Gigabyte, Terabyte, even Petabyte). Unlike being ignored or even considered useless before, in the last decades, the big data generating from producing and trading is considered as a treasure with of the development of data mining technology. As a result, the big data services which processes the big data are also brought to the fore now. An example of big data service is the customer mining service, which finds out those potential customers by analyzing the customers' purchasing behavior.

The data accessed by big data services are typically very large, which brings a big challenge to classical service selection methods. In the above example of customer mining service, all the known customer records should be read at the input, which can be as large as a few TB. When invoking these big data services in a composite service, there are huge data should be transmitted in the network. The time cost in transmission process is the transmission cost, which is not fully taken into account in classical service selection approaches. Most classical service selection approaches use the response time to identify the transmission cost, which is not suitable in this context now. Because the response time represents the time recorded from user calls until the service responses. It is usually much smaller than the transmission time; especially the size of data is further huge.

In order to select the optimal services from candidate services with transmission time of big data considered, the service location is introduced in this paper. The location of a service is determined by the area the service belongs to. The area is an abstract concept which is defined as a set of services with high speed communication connected. That is to say, a sub network, a cloud, even a computer is suitable to be considered as an area. The transmission cost among two services in a same area is small enough to be ignored when comparing with those deployed in two different areas. By considering these definitions and assumptions, the optimal service selection turns out to be an optimization problem that how to select the most concentrated services.

It is quite difficult to solve this optimization problem. However, it is considerable to reduce the original problem to a shortest path selecting problem. An algorithm based on shortest path selection is proposed to solve the original optimization problem.

In order to give a better understanding of our model and algorithm, we discussed a fully service selection example about railway disease detection. The railway disease detection is a daily work to check the status of each part of the railway to find out the potential damage and broken (called disease). We have studied the data of JGX (Beijing-Guangzhou railway line, China) which is as huge as 200MB per day. Three steps of analysis is needed for the collected raw data, namely Noise Reduction, Disease Detection and Classification. Lots of service can be used to handle each analysis step. Our job is to select the optimal services with less transmission cost.

The contribution of this paper can be stated as follows:

http://hipore.com/ijsc
• A location-aware service selection model. Differing from classical service selection approaches, the service location is introduced as a critical feature to identify the transmission cost for big data services and the optimal selection problem is converted into an optimization problem.

• An optimal service selection algorithm. An algorithm based on the shortest path selection for location aware service selection is proposed. The algorithm has a linear time complexity and easy to implement.

The rest of the paper is organized as follows. Section 2 reviews the classical service selection approaches. Section 3 proposes the system model with the location definition and a few considerable assumptions. Section 4 discusses the problem of location aware service selection and introducing the service selection algorithm. The case study is presented in section 5. The performance study is presented in section 6. Finally, section 7 concludes this paper and an outlook on possible continuations of our work.

2. RELATED WORK

Service composition is a classical problem and there are amounts of papers in this field. Zeng, L. (2003, 2004) proposes the composition method with the quality of service (QoS) considered. The idea caused lots of attentions. Alrifai, M. (2009, 2010a) converts the service composition problem to a decision problem on which component services should be selected such that user’s end-to-end QoS requirements (e.g., Availability, response time) and preference (e.g., Price) are satisfied. It is the source of selecting by solving the optimization problem. Mixed integer programming (MIP) is used which is quite widely used in this model. Zhang, M. (2010) takes the service environment into consideration. The black-box analysis method of optimizing composite service is adopted. Klein, A. (2010) considers repeated executions of services in the long-term. The modified problem is modeled with linear programming. And it is solved optimally in polynomial time. A distributed heuristic approach is proposed in Jing, L. (2010). Bakhshi, M. (2010) proposes an approach using fuzzy logic to infer based on quality measures ranked by user. Lecue, F. (2011) takes the semantic dimension into consideration. Jin, J. (2011) proposes a heuristic service composition method, named LOEM-T (Local Optimization and Enumeration Method with Solution Tendency Estimation). Wagner, F. (2011) utilizes a data structure which arranges functionally similar services in clusters and computes the QoS of each cluster. This idea is quite interesting and it benefits a lot in composition. There are more composition methods including Ardagna, D. (2007), Funk, C. (2007), Li, H (2011), Bo, Y. (2011), Babamir, S. (2011) and Wang, Pengwei (2011).

The QoS impacts the composition in two directions: service discovery (including recommendation) and service selection. Ran, S (2003) proposes a model for discovery in which functional and non-functional requirements are considered to evaluate QoS metrics. A metadata model on the basis of extended UDDI is proposed, where quality information data is used to describe the QoS of registered services including quality name, type, value, units, etc. Al-Masri, E. (2007) introduces a solution for controlling the discovery process across accessible services and Web Service Relevancy Function (WSRF) is used for measuring the relevancy ranking of a particular web service based on QoS metrics and client preferences. Mohana, R. (2011a, 2011b) presents an algorithm for building a rule-based model for ranking the service based on QoS using fuzzy clustering and particle swarm optimization (PSO). Paulraj, D. (2012) introduces OWL-S as a web ontology language for service discovery and composition. A service recommendation based on collaborative filtering is proposed in Tang, M. (2012).

There are lots of research achievements in service selection. Maximilien, E. (2003) proposes an approach in which agents assisting an application in selecting implementations that best match the quality criteria. Liu, Y. (2004) mentions that the QoS values cannot solely be collected from the service provider. Since this is subject to manipulation by the providers. In their framework, the QoS model is extensible and QoS information can be computed based on execution monitoring by users, or via requesters feedback. Soydan, B. (2004) proposes a framework combined an ontology of attributes with evaluation data. Ardagna, D. (2005) extends the mixed linear programming model to include local constraints and global constraints. Vu, L. (2005) presents a new QoS-based semantic web service selection and ranking solution with the application of a trust and reputation management method. Yu, T. (2005a, 2005b) models the end-to-end delay constraint as the multiple choice knapsack problem (MCKP) and provided efficient solutions. Cardellini (2007) proposes a method considering a group of request. A selection is carried out per group of requests rather than per-request. Kritikos (2009) has developed an extensible and rich ontology language for QoS-based WS description. Sun, Q. (2010a) proposes a quick service selection approach (QSSA) which adopts particle swarm optimization and fuzzy logic control to support fast and dynamic service selection. Kun, Z. (2010) introduces a composite agent service selection algorithm for non-functional attributes based on simulated annealing. Sun, Q. (2010b) proposes a new approach based on the notion of the skyline (SWS). Alrifai, M. (2010b) proposes a method quite similar to Sun, Q. (2010b). Selecting service from a set of functionally equivalent services is a multi-criteria decision making problem. Wang, Ping. (2011) takes the opinion that different consumers invariably hold differing views of the service contents and it is necessary to estimate the degree of consumer trust in a particular service based on the consumers’ direct experiment and indirect recommendation of the service. Suleiman, B. (2011) classifies consumers into groups/classes and optimized

These QoS-based selection methods solve the problem of selecting the optimal services for composition. However, these methods cannot solve the problem of location aware selection because the location has two important characteristics differing with the qualities mentioned in classical selection methods.

1. **Location is not quantitative.** A location of a service is not directly computable. The quantitative value extends from the location is the distance. The distance of any two services is determined when the locations of the two services are confirmed. It is better to understand distance as a relative value. Unlike location, other qualities such as reputation are really quantitative and it is an absolute value.

2. **Distance is context-sensitive.** There is not an optimal choice without considering the services selected before and after. The lowest transmission cost happens only with all nearby services have the smallest distance. Other qualities are not context-sensitive; it is possible to find an optimal service without considering the service before and after. Take the reputation as an example, the optimal service is the one with largest reputation.

Because of these different features, it is not possible to use the classical methods to solve the location aware selection problem. Furthermore, modeling of location-aware service selection was never formulated nor studied.

3. **MODEL**

3.1 **SERVICE SELECTION**

3.1.1 **BASIC SERVICE SELECTION**

Service selection is a critical part of service composition. Generally speaking, the process of service composition can be divided into three steps.

- **Building the composite process with abstract services.** Unlike the concrete service, the abstract service is the symbol representing a group of services with similar functions and interfaces. The abstract services are composed together by some control statements (such as assignment, switch and loop).

- **Finding some suitable services (namely candidate services) for each abstract service.** These candidate services have the same functions and interfaces as the abstract service. This step is known as service discovery. The discovery process is mainly searching first k candidate services for each abstract service in the service library with some functional constraints. Service recommendation is also similar, which selects the most suitable services and recommends to users.

- **Selecting an optimal service from each candidate service group.** The abstract services should be replaced with selected concrete ones in the composite process. This step is known as service selection. How to select...
the optimal services with location considered is the major problem discussed in this paper.

Fig. 1 presents the process of service discovery and service selection. In this figure, the composite service process is presented at the top, which fetches data from a database, calls three abstract services (present as the ellipses) and writes the result into another database. After the step of service discovery, alternative services are picked up from service library and constructed as candidate service groups. The last step is selecting the optimal concrete services from each candidate service group.

In most composite services, there are some control process patterns, as sequential process, parallel process, conditional process and loop process. Fig.2 represents the different control processes. (1) is the sequential process, (2) is parallel process, (3) is conditional process and (4) is loop process. The sample of Fig. 1 contains a parallel process. However, the hybrid control processes make it too complex to analyses the optimal selection. To simplify the problem, the sequential process is majority discussed in the rest, and the analysis of parallel process, conditional process and loop process will be extended in the future work. An example of sequential process is present in Fig. 3. In order to describe the following analysis precisely, some math symbols are introduced as follows.

A symbol $S$ represents the whole services used in selection, which is constructed by $M$ groups of candidate services. $S = \{S_1, S_2, ..., S_M\}$. Each $S_i$ not only represents an abstract service but also represents a group of candidate services having the same function as the abstract one. $S_i = \{S_{i1}, S_{i2}, ..., S_{iN}\}$. $N$ is the number of candidate services of each group. $S$ can also be represented by a matrix.

$$S = \begin{pmatrix}
    s_{11} & s_{12} & \cdots & s_{1n} \\
    s_{21} & s_{22} & \cdots & s_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_{m1} & s_{m2} & \cdots & s_{mn}
\end{pmatrix}$$  \hspace{1cm} (1)

There are two special services, namely $S_0$ and $S'_0$. They are the services providing the reading and writing operation of the database. $S_0$ and $S'_0$ should be confirmed before selecting.

### 3.1.2 Transmission Cost

The transmission cost is an important indicator of composite service. In the sequential process, the transmission cost happens since it spends some time transport the data from one service to another (the next one). A symbol $C_{ij} = T_{ij} \ast D_{ij}$ is used to represent the transmission cost from service $s_i$ to service $s_j$. $T_{ij}$ is the transmission speed from service $s_i$ to service $s_j$ and $D_{ij}$ is the amount of data. While $T_{ij}$ is determined by the network condition and the $D_{ij}$ is determined by composite process. The whole transmission cost is $C = \sum_{i=1}^{M-1} C_{i+1}$. It is easy to find out that $D_{ij}$ cannot be changed by the selection of service. The only way to minimize $D_{ij}$ is to modify the service process which is out of discussion of this paper. However, the different selection really determines the different $T_{ij}$. So the purpose of our model is to find out a group of optimal service selection minimize $T_{ij}$.

Fig. 4 represents a sample composite service with $S_1$, $S_2$ and $S_3$. The network condition determines $T_{112}$ and $T_{23}$. The whole transmission cost $C = C_{123} + C_{23} = T_{112} + T_{23} + D_{ij}$.  

### 3.2 Service Location

#### 3.2.1 Distribution Assumption

The location of a service is the feature representing its network environment. Generally speaking, the distribution of services is not random in the network. A reasonable assumption of the service distribution is proposed.

Assumption I. (The gathering): Services are not randomly distributed in the network. A certain number of services are naturally gathered as a service set. The services in one set are high speed connected. This services set is denoted as an area.

The Assumption I can be explained as follows. On one hand, the services provided by the same provider are deployed in the same server or some servers in the same subnet. In this situation, the server and the subnet is an area. On the other hand, a Cloud is an area with lots of services deployed on it by different providers. By our definition, a server is an area, a subnet is an area and even a Cloud is also an area. The only constraint is that the services in the same area are high speed connected. Fig. 5 presents the relation of areas and services. In Fig. 5, $S_1$, $S_2$, $S_3$, $S_4$, $S_5$ represent services. Area1, Area2 and Area3 represent areas. $S_1$, $S_2$, $S_3$ belong to Area1, $S_4$ belongs to Area2 and $S_5$ belongs to Area3. By considering the definition of area, $S_1$ and $S_2$ are high speed connect, namely the cost of transmitting data between them is quite low.
3.2.2 LOCATION

Considering a context with \( \mathcal{O} \) areas, \( A = \{A_1, A_2, ..., A_O\} \) is used to represent the all areas. The map from the services \( \mathcal{S} \) to the area \( A \) is confirmed before selection. In other words, the relationship of belonging is confirmed for any services. This belonging relationship represents the location of the service. A function (denotes \( \tau \)) is defined to identify the index of the area which the service belongs to.

\[
\tau(s) = \sum_{i=0}^{O} i \times \{s \in A_i\}
\]  

A special index function is used above, whose value equals 1 when the proposition in the braces is TRUE, and otherwise it equals 0:

\[
\{p : boolean\} = \begin{cases} 
1 & p \text{ is True} \\
0 & p \text{ is False} 
\end{cases}
\]  

The location of a service is represented by the area it belongs to, in other words, the index of its area, namely \( \tau(s) \).

3.3 SERVICE DISTANCE

3.3.1 DISTANCE OF AREAS

The distance of the two areas is the measurement of the speed of transmission data between the two areas. Since all the areas in the context are confirmed, the distances of each two of them are also confirmed. A matrix \( D \) is defined to represent these distances. \( D_{ij} \) represents the distance between \( A_i \) and \( A_j \).

Table 1 is an example of \( D \) representing the distances of areas in Fig. 5. In fact \( D \) is just the Distance matrix of the graph which constructed by the areas. Generally speaking, \( D_{ij} \neq D_{ji} \). Because the network transmission sometimes (differing from types of network) spends different time in uploading and downloading. The different types of network is not the majority of this paper, so it is considerable to generally use \( D \) to represent the distances rather than ensure \( D \) is a symmetrical matrix. While in order to simplify our computing, \( D_{ij} = D_{ji} \) is assumed in the following samples. For the convenience of writing, a function \( D(i, j) \) is defined instead of using \( D_{ij} \) directly.

**Table 1: An example of \( D \)**

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0</td>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>A2</td>
<td>D1</td>
<td>0</td>
<td>D3</td>
</tr>
<tr>
<td>A3</td>
<td>D2</td>
<td>D3</td>
<td>0</td>
</tr>
</tbody>
</table>

3.3.2 DISTANCE OF SERVICES

In order to give a better formalization we do not use the \( T_{r_{ij}} \) to represent the basic indicator of services, service distance is defined as the mean transmission speed of two services. We use \( d_{ij} \) to denote the distance between service \( s_i \) and service \( s_j \); \( d_{ij} \) is the mean of \( T_{r_{ij}} \) through a period of time. \( T_{r_{ij}} \) changes in different time. Since in the rush-hour, \( T_{r_{ij}} \) is much smaller than that in other time. Here is a proposition about service distance.

**Proposition II (Service Distance):** For any service \( s \) and \( d_i \) represents the distance between service \( s \) and service \( s_j \), which is in the same area as \( s \) and \( d_i \) represents the distances between service \( s \) and service \( s_j \), which is in another area, we have \( d_i < d_j \).

**Proof.** Let’s assume that existing a \( i \) and a \( j \), such that \( d_i > d_j \). It contradicts with the definition of area. Because the Assumption 1 guarantees that the services in the same area have a high speed connection. If \( d_i > d_j \), it is the service \( s_j \) that in the same area with \( s \) instead of service \( s_i \). It contradicts the condition above.

If exist a \( j \) and a \( i \), such that \( d_i = d_j \), \( s_j \) and \( s_i \) should either both in the same area with \( s \) or both not in.

http://hipore.com/ijsc 56
It contradicts assumptions that service $s_i$ is in another area. Q.E.D.

Proposition II guarantees $d_i < d_j$, which is a weak condition. Following assumption provides a strong condition.

**Assumption III (Strong Service Distance Condition):**
For any service $s_i$, $d_i$ represents the distance between service $s$ and service $s_i$, which is in the same area as $s$ and $d_j$ represents the distances between service $s$ and service $s_j$, which is in another area, we have $d_i < d_j$.\[i, j\]

The Assumption III is easy to understand because the transmission cost between the two services in a same subnet is always much smaller than the cost crossing subnets. The strong condition makes it possible to define the distance of any two services in a simple way.

The distance of any two services is defined by the distance of the areas they each belongs to. For service $s_{ij}$ and $s_{kj}$ the distance function $\text{dis}(s_{ij}, s_{kj})$ is defined as follows:

$$\text{dis}(s_{ij}, s_{kj}) = D(I(s_{ij}), I(s_{kj}))$$

(4)

Since the strong condition, 0 is used to represent the distance between services in the same area. The distance of $S1$ and $S4$ is just $D1$ in Figure 5, while the distance of $S1$ and $S3$ is 0.

**4. LOCATION AWARE SELECTION**

**4.1 EVALUATION FUNCTION**

With the definition of distance under the assumptions mentioned in section 3, the problem of selecting optimal services with lowest transmission cost can be converted into a problem of selecting optimal services with the smallest distance between each two services. Considering the sequential process described in Fig. 6 and the candidate services $S$ defined in Formula (1), the vector $\theta = (\theta_1, \theta_2, ..., \theta_M)$ is used to identify a considerable selected service group. For an element $\theta_i$ in $\theta$, the service $s_{i\theta_i}$ is selected as the concrete service for the abstract $S_i$, \(\omega_i\) is the weight coefficient, which is also a vector. The sum of the distances of the selected group $\theta$ is defined as an evaluation function as follows.

$$F(\theta) = \sum_{i=1}^{m-1} \text{dis}(s_{i\theta_i}, s_{(i+1)\theta_{i+1}}) \omega_i$$

(5)

Considering the distance between the first service and the database service $S_0$ and the distance between the last service and the database service $S'_0$, the complete evaluation function is defined as follows.

$$F'(\theta) = \text{dis}(s_0, s_{\theta_1}) \omega_0 + F(\theta) + \sum_{i=1}^{m-1} \text{dis}(s_{i\theta_i}, s'_0) \omega_m$$

(6)

The Simple Additive Weighting (SAW) is used here. The \(\omega_i\) is the weight coefficient. The weight coefficient really makes sense because the transmission cost is not only related to the transmission speed but also related to the size of data. Generally speaking, the size of data in different part of composite process is different. An example is that $S_1$ read the whole data from the database service $S_0$ as the input, while output the statistic result of the origin data. The second service $S_2$ reads the statistic result as the input. In this condition, the distance between $S_1$ and $S_0$ has more influence on the whole transmission cost than that between $S_1$ and $S_2$. So we can initialize the weight parameter with $\omega_1 > \omega_2$.

**4.2 OPTIMIZATION PROBLEM**

Finding the smallest $F'$ with a feasible parameter $\theta$ is an optimization problem, which can be described as follows:

$$\begin{align*}
\text{inf} & \quad F'(\theta) \\
\text{subject to} & \quad \theta_i \in [1, N] \\
& \quad \theta_i \in \mathbb{Z}
\end{align*}$$

(7)

This formula means the feasible set of parameter $\theta$ is constrained by $1 \leq \theta_i \leq N$ and $\theta_i$ is an integer. The optimal solution $\hat{\theta}$ satisfies $\hat{\theta}$ belongs to feasible set and $\forall \theta, F'(\theta) \leq F'(\hat{\theta})$. Following theorem presents the relationship with this optimization problem with our original service selection problem.

**Theorem IV** (Optimization problem): A location aware service selection problem with service distribution $I$ and area distance $D$ confirmed, is equivalent to the optimization problem described in (7).

The proof of this theorem is omitted because it is mentioned in previous sections. More concern is given on the solving of this optimization problem.

Formula (7) is an integer programming (IP) problem, which is a famous NP problem. A problem belonging to NP class means its positive solutions can be found in polynomial time on a non-deterministic machine. Generally speaking, there is not an algorithm with Non-deterministic Polynomial time complexity [21]. A famous IP problem is Traveling Salesman Problem (TSP): Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and
returns to the origin city?[22] However, because of the features of our model, it is possible to convert our problem to a shortest path problem of a special graph and an algorithm of polynomial time complexity is proposed later.

4.3 Shortest Path Problem

A possible way of solving problem (7) is iterating with all the possible combination tried and finding the combination with \( F'(\theta) \) minimized, called the iterating algorithm. Iterating algorithm works in this way: iterating the whole possible combination, which is as much as \( M^N \) to get the best selection (minimized \( F \)). It is easy to find that iterating algorithm reaches the global optimal solution with time complexity \( O(M^N) \), which is exponential and not useful in practice.

In order to fully take advantage of the structural features, the optimization problem (7) is converted into a conditional shortest path selection problem. The transferring process can be described as the following steps.

1. Iterating all candidate services (including database services \( S'_0 \) and \( S'_0 \)) and doing step 2 and 3.
2. Append service \( s_i \) in the vertex set \( V \).
3. Iterating all candidate services in the next group. For each \( s_{i+1,k} \), Append edge \( E_{ijk} \) which is the directed edge connecting from the service \( s_i \) to the service \( s_{i+1,k} \) in edge set \( E \). The length of the edge \( E_{ijk} = \text{dis}(s_{ij}, s_{i+1})\lambda_c \).
4. Find the shortest path from \( S_0 \) to \( S'_0 \).

The shortest path problem is a classical topic which has been studied in many researches. Once we can guarantee that the vertices of the shortest path are the optimal selection.

**Lemma V.** The number of vertices of a path from \( S_0 \) to \( S'_0 \) is \( M+2 \). The \( i \)-th vertex is one of services in \( S_1 \).

**Proof.** The first conclusion is natural because the transferring process described above. If exist a \( k \), that the \( k \)-th vertex does not belong to \( S_1 \) then \( (i + 1) \)-th vertex must does not belong to \( S_i \). Because, the edges to vertex in \( S_i \) comes from \( S_{i-1} \). We can reduce that \( (M + 2) \)-th vertex is not \( S'_0 \) which contradicts with the first conclusion. Q.E.D.

With the lemma, it comes to a useful theorem which guarantees the optimal selection.

**Theorem VI** (Shortest path). The vertices of the shortest path from \( S_0 \) to \( S'_0 \) is the optimal solution of the optimization problem (7).

**Proof.** Sufficiency: with the help of Lemma V, it is easy to find that for the shortest path \( \{v_1, v_2, ..., v_t\} \) there are three important facts:

1. The number of vertices of shortest path is equal to \( M+2 \) to the whole number of abstract services and database services, namely \( t = M + 2 \).
2. \( v_i \) is \( S_0 \) and \( v_{M+2} \) is \( S'_0 \).
3. For any \( i \in [2, M+1] \), \( v_i \) is a service belongs to candidate group \( S_{i-1} \).

These facts guarantee that \( \{v_1, v_2, ..., v_t\} \) represents a services selection \( \{S_1, S_2, ..., S_M\} \) with an evaluation value \( F' \). The selection is also optimal. Assuming that there is another selection \( \{S_1, S_2, ..., S_M\} \) with \( F' \) and \( F' > F' \). Let \( \{v_1, v_2, ..., v_t\} \) be the corresponding vertices of \( \{S_1, S_2, ..., S_M\} \). Then the path length of \( \{v_1, v_2, ..., v_t\} \) must be smaller than which of \( \{v_1, v_2, ..., v_t\} \) and this is inconsistent with the fact that \( \{v_1, v_2, ..., v_t\} \) is the shortest path. So the assumption is not true and the shortest path guarantees the optimal solution of optimization problem (7).

Necessity: the necessity represents that if the optimal result of optimization problem (7) is known, denotes \( \{S_1, S_2, ..., S_M\} \) its corresponding path is the shortest path. The necessity can be proved by constructing the path with few steps:

1. Let \( v_1 \) be \( S_0 \) and \( v_{M+2} \) be \( S'_0 \).
2. Let \( v_i \) be \( S_{i-1} \).

The path of \( \{v_1, v_2, ..., v_t\} \) is the shortest path. Q.E.D.

The Theorem guarantees the equation of the optimization problem (7) and the graph shortest path problem.

4.4 Algorithm

With the help of Theorem VI, the optimization problem can be convert into a shortest path problem without any generality. The graph shortest path problem is quite classical problem and lots of algorithms have been proposed in last thirty years. Bellman-Ford algorithm solves the single-source problem if edge weights may be negative [23]. A* search algorithm uses heuristics to try to speed up the search [24]. Dijkstra’s algorithm [25] is the most widely used algorithm solving the single-source shortest path problems, and it adopts different kinds of situation and easy to carry out. So it is used in our solution.

Our full algorithm is presented in Fig. 7. The algorithm is named LA-selection (Location Aware selection). In this algorithm, \( M \) is the groups of candidate services, \( N \) is the number of candidate service each group. \( S \) stores not only the candidate services but also the two database services. The members in the first row of \( S \) are both \( S_0 \) and the members in the last row are both \( S'_0 \). \( D \) stores the distances of each area and \( \Omega \) is the weight coefficient.

The first part of the algorithm converts \( S \) to a vertex set \( V \). A considerable improvement is for any group contains two or more services in one area, any of them is feasible. So it is better to use only one vertex to represent these services as the code in line 4. The improvement reduces the number of vertices and promote the efficiency.

The second part of the algorithm is the Dijkstra’s Algorithm. Once the last vertex (namely \( S'_0 \)) is reached in the iteration, the loop stops.

The last part is generating the optimal selection as theta.
Algorithm 1: LA_Selection(S, N, D, Omega)

Input: M groups candidate services S, N is the number of services each group, D is the distance matrix and Omega is the weight coefficient

1. k=0;
2. for i = 1:M+2
3.     for j = 1:N
4.         if not ExistSameArea(S[i,j])
5.             V[k] = S[i,j]; //initialing the vertex set V
6.             V[k].j = j;
7.             V[k].dist = MAX; //initialing distance
8.             k++;
9.     end
10.    end
11. end
12. V[1].dist = 0;
13. si = FindMinDist(V);
14. while si > 0
15.     if IsLast(V[si]) //if V[si] is the last
16.         F = V[si].dist; //F is the shortest distance
17.     Break;
18. else
19.     for i iterats next group
20.         if V[i].dist > V[si].dist + Omega[si] * D(Area(V[si]), Area(V[i]))
21.             V[i].dist = V[si].dist + Omega[si] * D(Area(V[si]), Area(V[i]))
22.         end
23.     end
24.     end
25. end
26. si = FindMinDist(V);
27. end
28. for i = 1:M-1:0 //generating the optimal selection as theta.
29.     theta[i] = V[si].j;
30.     si = V[si].last;
31. end

Figure 7: Location Aware Selection Algorithm

4.5 FRAMEWORK

With all the preparation theorems, we now can propose the location aware service selecting framework.

Finding out the areas of each candidate services belong to and the distances between each area. These information can be collected by two ways. The first is the information of service provider. It is natural to find that the services of the same provider may be in the same area. The simple way of division of area is classifying services by their providers. The second is the logs of services. Most logs contains the calling time and response time. It is possible to estimate the distance of areas by these logs. In fact it is quite complex problem to estimate the distance of areas and this is not the majority of this paper.

Allocating the value of weight coefficients \( \omega_e \). The weight coefficient is proportional to the size of data. \( \omega_e \propto D_{A_{e+1}} \). It is determined by the design of service process which is available before the service selection.

Converting original process into a graph. The vertices are corresponding to candidate services. The edges connect all the services from one candidate group to each services in next candidate services. This step has been fully discussed in section 4.3.

Finding out the shortest path from \( S_0 \) to \( S'_0 \). The algorithm 1 in Fig.7 is used in this step.

Because of Theorem VI, the service invoking in the shortest path from \( S_0 \) to \( S'_0 \) is the optimal solution \( \hat{\theta} \). The Theorem IV guarantees that the optimal solution \( \hat{\theta} \) is the optimal selection with lowest transmission cost. Fig. 8 represents an example of converting service selection problem to the shortest path problem. There are three candidate groups (M=3) in this example and each group have 3 services (N=3). The graph is represent in the top right of Fig. 8. It is exactly a full connect graph for each two candidate service group. And the full graph is the n partite graph. The vertices in the shortest path are bold in the right bottom part of Fig. 8. The correspondiong services (namely the optimal selection services) are bold in the left bottom part of Fig. 8.

5. CASE STUDY

5.1 INTRODUCTION

In order to understand our model, an example of railway status detection is studied. The railway status detection is a daily work to check the status of each part of the railway to find out the potential disease (the damage and the broken of...
the railway segment). Daily railway status detection ensures the safety of the railway.

We have studied the data of JGX (Beijing-Guangzhou railway line, China). The detecting train collects data (as a record) 4 times per meter. 24 kinds of data are recorded in a record. They are Meters, Flags, Event, Lprf (mm), Rprf (mm), Lahn (mm), Rahn (mm), Gage (mm), Can’t (mm), Xlvl (mm) and etc. The whole length of JGX is 2300 kilometers. There are 2300*1000*4 = 9200000 records are collected in once detection. The size of data in once collection is nearly 200MB. The frequency of detection varies in different seasons. However the size of data collected a month is nearly 12GB. The whole railway detection data are nearly 450GB.

The data were analyzed manually in the past. It took much human labor. Recently some companies have started to develop professional data analyzing service for the railway department while different service providers are expert in different domain, e.g. some have provided Data filtering services, some have provided Data Encryption services, etc. It is necessary to compose these services together to fit the fully requirement. Railway status detection usually has these steps.

- Collection. Detecting the railway by a special detecting train, collecting variety kind of data, including gauges, warps and etc. This data is as large as a few Gigabytes for a long railway. This step does not belong to data analysis and it can only be done by the railway department itself.
- Noise Reduction. Original data contain many noises caused by the measurement error. It is necessary to reduce these noises before deep analysis. This service is \( S_1 \).
- Disease Detection. Analyzing the collected data to find out the potential diseases (the disease means the damage or broken in railway segment), by comparing the data with some special patterns of potential diseases. This service is \( S_2 \).
- Classification. Different diseases may cause different problems. Some diseases can be ignored, while some diseases may even cause derailment. So it is necessary to classify these diseases into different levels. This service is \( S_3 \).

In this example, \( S_0 \) is the service providing database containing the collected data. \( S_0' \) is the service maintaining database containing different levels of diseases. In reality, \( S_0 \) and \( S_0' \) are usually the same. But we still assume that they are different without loss of generality. The whole composite process of railway detection is represented in Fig. 9.
The value of $\omega_i$ is decided by the size of data transmitted between services. In the sample of railway detection, data transferring from $S_0$ to $S_1$ are the original data. The data transferring from $S_1$ to $S_2$ have the same size as the original data. The data used in $S_3$ and written into $S_0'$ is just the data of disease points, whose size is only near 0.01% of the whole data. A possible allocation of $\omega_i$ is $\omega_i = \omega_1 = 10000$, $\omega_i = \omega_2 = 1$.

5.4 CONVERSION

It is needed to convert original sequential process into a graph. The vertices of the graph are the candidate services. The edges are the Connection of service $s_{ij}$ with each candidate service $s_{i+1,k}$ in the next candidate service group. The length of the edge is $\text{dis}(s_{ij}, s_{i+1,k}) \omega_t$. There are 24 edges: $(s_0, s_1), (s_0, s_1), (s_0, s_1), (s_0, s_1), (s_1, s_2), (s_1, s_2), (s_1, s_2), (s_1, s_2), (s_2, s_3), (s_2, s_3), (s_2, s_3), (s_2, s_3), (s_3, s_4), (s_3, s_4), (s_3, s_4), (s_3, s_4), (s_4, s_5), (s_4, s_5), (s_4, s_5), (s_4, s_5), (s_5, s_6), (s_5, s_6), (s_5, s_6), (s_5, s_6)$. The distance matrix the graph is described in follows in Table 4, in which, N represents unreachable.

<table>
<thead>
<tr>
<th></th>
<th>S0</th>
<th>S11</th>
<th>S12</th>
<th>S13</th>
<th>S21</th>
<th>S22</th>
<th>S23</th>
<th>S31</th>
<th>S32</th>
<th>S33</th>
<th>S0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>N</td>
<td>0</td>
<td>120000</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>S11</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>120000</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>S12</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0</td>
<td>220000</td>
<td>2000000</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>S13</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>120000</td>
<td>240000</td>
<td>4000000</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>S21</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>S22</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>432</td>
<td>432</td>
<td>22</td>
</tr>
<tr>
<td>S23</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>S31</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>200</td>
</tr>
<tr>
<td>S32</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>200</td>
</tr>
<tr>
<td>S33</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

5.5 FINDING SHORTEST PATH

The algorithm is presented in Fig. 7. The M here is 3 and N is also 3. The shortest path is $s_0 \rightarrow s_{11} \rightarrow s_{21} \rightarrow s_{31} \rightarrow s_0'$. The shortest distance from vertex $S_0$ to $S_0'$ is 120012. So the optimal choice plan is selecting the first service in group1, the first service in group2 and the third service in group3.

6. PERFORMANCE STUDY

The time complexity of LA selection algorithm is $O(MN^2)$, where $M$ is the number of the groups of candidate services, namely, the number of rows in Formula (1), and $N$ is the number of candidate services each group, namely the number of columns in Formula (1). In most conditions the number of candidate services is confirmed, which is decided by the service discovery. Therefore, the selection time of LA is only determined by the number of the groups of candidate services (namely $M$). The time complexity of LA selection is $O(M)$, which means it is linear time complexity.

We compared the time complexity of our algorithm with the original iteration algorithm mentioned in section 4.3. In order to consider the time complexity of the different sequential process, we evaluated the calculating with $M$ changes from 5 to 5000. At the same time, The number of candidate services each group (namely $N$), the number of areas and the distance between each two areas are fixed. The comparison result is presented in Fig. 10. The x-axis is $M$, which is from 5 to 5000 and the y-axis is the logarithm $log(T)$, where

$$log(T) \approx 4 - \frac{1}{3}log(M) - \frac{1}{2}log(N) - \frac{1}{4}log(N^2)$$.

Figure 10 Comparison of algorithms

http://hipore.com/ijsc
of time. The dash line represents the iteration algorithm. It is exponential time complexity. The solid line represents our algorithm and it is linear time complexity.

7. CONCLUSION AND FUTURE WORK

In this paper, the service location is introduced as a new feature helping to select the optimal services with lowest transmission cost. The classical service selection methods mentioned in [1-3, 5] concerned with the general QoS evaluation in service selection. Data transmission cost is ignored by the classical service selection methods. However, the era of big data brings a new challenge for service selection. The huge size of data makes the transmission time been a majority challenge. In this context, a service selection model with transmission time concerned is needed, which is also the motivation of our research. Location is introduced to describe the network context of a service with two distribution assumptions mentioned in section 3.2. The distance of any two services is introduced to represent the transmission speed between these two services. The evaluation function of service selection is the Simple Additive Weighting (SAW) of each distance between two nearby services in the sequential process. In order to solve this optimization problem, the original data are converted into a graph and the vertices invoked in the shortest path from the data reading service (namely $s_0$) to data writing service (namely $s'_0$), are the optimal selection of the original optimization problem, which is also the optimal selection. A fully case about the railway status of the original optimization problem, which is also the motivation of our research.

In this paper, the service location is introduced as a new feature helping to select the optimal services with lowest transmission cost. The classical service selection methods mentioned in [1-3, 5] concerned with the general QoS evaluation in service selection. Data transmission cost is ignored by the classical service selection methods. However, the era of big data brings a new challenge for service selection. The huge size of data makes the transmission time been a majority challenge. In this context, a service selection model with transmission time concerned is needed, which is also the motivation of our research. Location is introduced to describe the network context of a service with two distribution assumptions mentioned in section 3.2. The distance of any two services is introduced to represent the transmission speed between these two services. The evaluation function of service selection is the Simple Additive Weighting (SAW) of each distance between two nearby services in the sequential process. In order to solve this optimization problem, the original data are converted into a graph and the vertices invoked in the shortest path from the data reading service (namely $s_0$) to data writing service (namely $s'_0$), are the optimal selection of the original optimization problem, which is also the optimal selection. A fully case about the railway status detection is studied and the time complexity of our algorithm is analyzed. It is $O(M)$ with the number of candidate services each group confirmed. Our future work will mainly focus on unifying the location aware selection model with the classical QoS aware selection model to propose a more general one and extending our model to other control process, such as loop and parallel process.

8. ACKNOWLEDGMENT

This work was supported by National Science and Technology Supporting Program of China (No. 2013BAH10F02), National Natural Science Foundation of China under Grant (No.61272129) and Zhejiang Provincial Natural Science Foundation of China under grant No.LY12F02029

9. REFERENCES


Authors

Ying Li received the B.S., M.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1994, 1997 and 2000, respectively. He is currently an associate professor with the College of Computer Science, Zhejiang University, and a Visiting Professor with the University of California at Santa Barbara. He is currently leading some research projects supported by National Natural Science Foundation of China and National High-tech R&D Program of China (863 Program). His research interests include service computing, business process management and compiler.

Zhiling Luo is the Ph.D. student in College of Computer Science, Zhejiang University, China. He received his B.S. in Computer Science from Zhejiang University in 2012. His research interests include service computing, social network and data mining.

Jianwei Yin is currently a professor in the College of Computer Science, Zhejiang University, China. He received his Ph.D. in Computer Science from Zhejiang University in 2001. He is the visiting scholar of Georgia Institute of Technology, US, in 2008. His research interests include distributed network middleware, software architecture and information integration.
IEEE 7th International Conference on Cloud Computing (CLOUD 2014)


IEEE 11th International Conference on Services Computing (SCC 2014)

SCC 2014 will focus on services innovation lifecycle e.g., enterprise modeling, business consulting, solution creation, services orchestration, optimization, management, and BPM. Visit http://conferences.computer.org/scc/.

IEEE 21th International Conference on Web Services (ICWS 2014)


IEEE 3rd International Conference on Mobile Services (MS 2014)

MS 2014 will feature Wearable technology and applications. Topics cover but not limited to all innovative aspects of wearable devices, programming models, integration, and domain specific solutions. Visit http://themobileservices.org/2014.

IEEE 10th World Congress on Services (SERVICES 2014)

June 27—July 2, 2014, Anchorage, Alaska, USA
Federation of 5 Service-Centric Conferences from Different Angles (http://www.servicescongress.org)

International Congress on Big Data (BigData 2014)

BigData 2014 aims to explore various aspects of Big Data including modeling, storage architecture (NOSQL), enterprise transformation, text mining, social networks, applied analytics and various applications, Big Data As A Service. Visit http://ieeebigdata.org/2014.


Conference proceedings are EI indexed. Extended versions of invited ICWS/SCC/CLOUD/MS/BigData papers will be published in IEEE Transactions on Services Computing (TSC, SCI & EI indexed), International Journal of Web Services Research (JWSR, SCI & EI indexed), International Journal of Business Process Integration and Management (IJBPIM), IEEE IT Pro (SCI & EI Indexed).

Contact: Liang-Jie Zhang (LJ) zhanglj@ieee.org (Steering Committee Chair)
Submission Deadlines
ICWS 2014: 1/15/2014
CLOUD 2014: 1/15/2014
SCC 2014: 1/31/2014
MS 2014: 1/31/2014
BigData 2014: 2/15/2014
SERVICES 2014: 2/15/2014
Call for Articles
International Journal of Cloud Computing
ISSN (Print) 2326-7542, ISSN (Online) 2326-7550 http://www.hipore.com/ijcc/

Mission
Cloud Computing has become the *de facto* computing paradigm for Internet-scale service development, delivery, brokerage, and consumption in the era of Services Computing, fueling innovative business transformation and connected human society. 15 billion smart devices would be communicating dynamically over inter-connected clouds by 2015 as integral components of various industrial service ecosystems. The technical foundations of this trend include Service-Oriented Architecture (SOA), business & IT process automation, software-defined computing resources, elastic programming model & framework, and big data management and analytics. In terms of the delivered service capabilities, a cloud service could be, among other as-a-service types, an infrastructure service (managing compute, storage, and network resources), a platform service (provisioning generic or industry-specific programming API & runtime), a software application service (offering email-like ready-to-use application capabilities), a business process service (providing a managed process for, e.g., card payment), a mobile backend service (facilitating the integration between mobile apps and backend cloud storage and capabilities) or an Internet-of-things service (connecting smart machines with enablement capabilities for industrial clouds).

The International Journal of Cloud Computing (IJCC) aims to be a reputable resource providing leading technologies, development, ideas, and trends to an international readership of researchers and engineers in the field of Cloud Computing. IJCC only considers expanded version of papers presented at high quality conferences such as IEEE International Conference on Cloud Computing.

Topics
The International Journal of Cloud Computing (IJCC) covers state-of-the-art technologies and best practices of Cloud Computing, as well as emerging standards and research topics which would define the future of Cloud Computing. Topics of interest include, but are not limited to, the following:

- ROI Model for Infrastructure, Platform, Application, Business, Social, Mobile, and IoT Clouds
- Cloud Computing Architectures and Cloud Solution Design Patterns
- Self-service Cloud Portal, Business Dashboard, and Operations Management Dashboard
- Autonomic Process and Workflow Management in Clouds
- Cloud Service Registration, Composition, Federation, Bridging, and Bursting
- Cloud Orchestration, Scheduling, Autoprovisioning, and Autoscaling
- Cloud Enablement in Storage, Data, Messaging, Streaming, Search, Analytics, and Visualization
- Software-Defined Resource Virtualization, Composition, and Management for Cloud
- Security, Privacy, Compliance, SLA, and Risk Management for Public, Private, and Hybrid Clouds
- Cloud Quality Monitoring, Service Level Management, and Business Service Management
- Cloud Reliability, Availability, Serviceability, Performance, and Disaster Recovery Management
- Cloud Asset, Configuration, Software Patch, License, and Capacity Management
- Cloud DevOps, Image Lifecycle Management, and Migration
- Cloud Solution Benchmarking, Modeling, and Analytics
- High Performance Computing and Scientific Computing in Cloud
- Cloudlet, Cloud Edge Server, Cloud Gateway, and IoT Cloud Devices
- Cloud Programming Model, Paradigm, and Framework
- Cloud Metering, Rating, and Accounting
- Innovative Cloud Applications and Experiences
- Green Cloud Computing and Cloud Data Center Modularization
- Economic Model and Business Consulting for Cloud Computing
Call for Articles
International Journal of Big Data
ISSN: 2326-442X  http://www.hipore.com/ijbd/

Mission
Big Data has become a valuable resource and mechanism for the practitioners and researchers to explore the value of data sets in all kinds of business scenarios and scientific investigations. New computing platforms such as cloud computing, mobile Internet, social network are driving the innovations of big data. From government initiative perspective, Obama Administration in United States launched "Big Data" initiative that announces $200 Million in new R&D investments on March 29, 2012. European Union also announced "Big Data at your service" on July 25, 2012. From industry perspective, IBM, SAP, Oracle, Google, Microsoft, Yahoo, and other leading software and internet service companies have also launched their own innovation initiatives around big data.

The International Journal of Big Data (IJBD) aims to provide a reputable publication channel for all authors working in the field of all aspects of Big Data. Big Data is a dynamic discipline. One of the objectives of IJBD is to promote research accomplishments and new directions. Therefore, IJBD welcomes special issues in any emerging areas of big data.

Topics
IJBD includes topics related to the advancements in the state of the art standards and practices of Big Data, as well as emerging research topics which are going to define the future of Big Data. Topics of interest to include, but are not limited to, the following:

Big Data Models and Algorithms (Foundational Models for Big Data, Algorithms and Programming Techniques for Big Data Processing, Big Data Analytics and Metrics, Representation Formats for Multimedia Big Data)

Big Data Architectures (Cloud Computing Techniques for Big Data, Big Data as a Service, Big Data Open Platforms, Big Data in Mobile and Pervasive Computing)

Big Data Management (Big Data Persistence and Preservation, Big Data Quality and Provenance Control, Management Issues of Social Network enabled Big Data)

Big Data Protection, Integrity and Privacy (Models and Languages for Big Data Protection, Privacy Preserving Big Data Analytics Big Data Encryption)

Security Applications of Big Data (Anomaly Detection in Very Large Scale Systems, Collaborative Threat Detection using Big Data Analytics)

Big Data Search and Mining (Algorithms and Systems for Big Data Search, Distributed, and Peer-to-peer Search, Machine learning based on Big Data, Visualization Analytics for Big Data)
