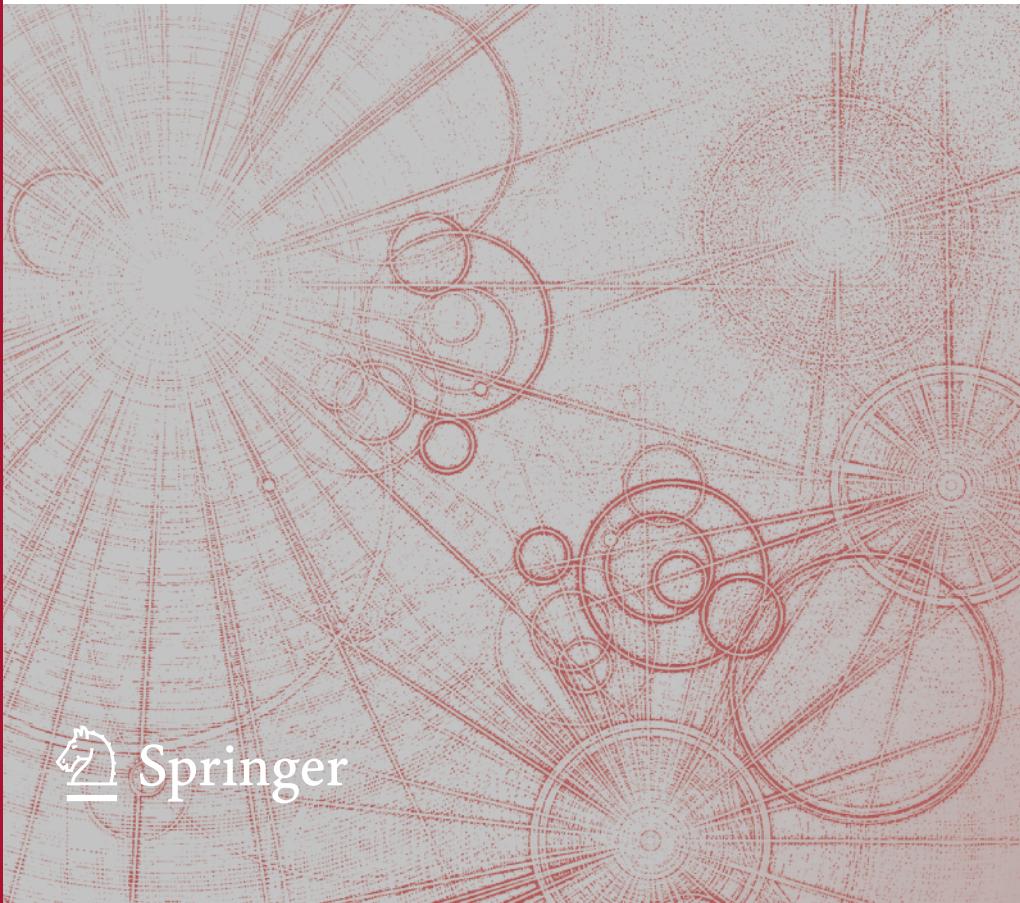


Lars Braubach · Juan M. Murillo
Nima Kaviani · Manuel Lama
Loli Burgueño · Naouel Moha
Marc Oriol (Eds.)

Service-Oriented Computing – ICSOC 2017 Workshops

ASOCA, ISyCC, WESOACS, and Satellite Events
Málaga, Spain, November 13–16, 2017
Revised Selected Papers



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison, UK

Takeo Kanade, USA

Josef Kittler, UK

Jon M. Kleinberg, USA

Friedemann Mattern, Switzerland

John C. Mitchell, USA

Moni Naor, Israel

C. Pandu Rangan, India

Bernhard Steffen, Germany

Demetri Terzopoulos, USA

Doug Tygar, USA

Gerhard Weikum, Germany

Services Science

Subline of Lecture Notes in Computer Science

Subline Editors-in-Chief

Athman Bouguettaya, *RMIT University, Melbourne, Australia*

Michael P. Papazoglou, *University of Tilburg, The Netherlands*

Subline Editorial Board

Boualem Bentallah, Australia

Paul Maglio, USA

Murthy Devarakonda, USA

Klaus Pohl, Germany

Carlo Ghezzi, Italy

Stefan Tai, Germany

Chi-Hung Chi, Tasmania

Yuzuru Tanaka, Japan

Hani Jamjoom, USA

Christopher Ward, USA

Ingolf Krueger, USA

More information about this series at <http://www.springer.com/series/7408>

Lars Braubach · Juan M. Murillo
Nima Kaviani · Manuel Lama
Loli Burgueño · Naouel Moha
Marc Oriol (Eds.)

Service-Oriented Computing – ICSOC 2017 Workshops

ASOCA, ISyCC, WESOACS, and Satellite Events
Málaga, Spain, November 13–16, 2017
Revised Selected Papers



Springer

Editors

Lars Braubach
Hochschule Bremen
Bremen
Germany

Juan M. Murillo 
University of Extremadura
Cáceres
Spain

Nima Kaviani
IBM Cloud Labs
San Jose, CA
USA

Manuel Lama 
University of Santiago de Compostela
Santiago de Compostela
Spain

Loli Burgueño 
University of Malaga
Málaga
Spain

Naouel Moha
University of Quebec at Montreal
Montréal, QC
Canada

Marc Oriol 
Universitat Politècnica de Catalunya
Barcelona
Spain

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-319-91763-4

ISBN 978-3-319-91764-1 (eBook)

<https://doi.org/10.1007/978-3-319-91764-1>

Library of Congress Control Number: 2018944398

LNCS Sublibrary: SL2 – Programming and Software Engineering

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume presents the proceedings of the scientific satellite events that were held in conjunction with the 2017 International Conference on Service-Oriented Computing, which took place in Málaga, Spain, November 13–16, 2017. Since the first edition in Trento in 2003, ICSOC has become a leading conference in the rapidly evolving areas of service research.

The satellite events provide venues for specialist groups to meet, to generate focused discussions on specific sub-areas within service-oriented computing, and to engage in community-building activities. These events helped significantly enrich the main conference by both expanding the scope of research topics and attracting participants from a wider community. The selected scientific satellite events were organized around three main tracks, including a workshop track, a PhD symposium track, and a demonstration track.

The interest in the ICSOCs 2017 workshop track was high and from eight workshop proposals three were finally selected. The workshop track included a wide range of topics that fall into the general area of service computing. A special focus this year was on cloud computing and the Internet of Things, also reflected in the titles of workshops. These technology trends in combination with novel application domains led to inspiring research results.

- ASOCA 2017: the Second Workshop on Adaptive Service-Oriented and Cloud Applications
- ISyCC 2017: the Second Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities
- WESOACS 2017: the 13th International Workshop on Engineering Service-Oriented Applications and Cloud Services.

The workshops were held on November 13, 2017. Each workshop had its own chairs and Program Committee who were responsible for the selection of papers. The overall organization for the workshop program, including the selection of the workshop proposals, was carried out by Lars Braubach and Juan M. Murillo. The ICSOC PhD Symposium is an international forum for PhD students to present, share, and discuss their research in a constructive and critical atmosphere. It also provides students with fruitful feedback and advice on their research approach and thesis. The PhD Symposium Track was chaired by Loli Burgeño and Naouel Moha.

The ICSOC Demonstration Track offers an exciting and highly interactive way to show research prototypes/work in service-oriented computing (SOC) and related areas. The Demonstration Track was chaired by Nima Kaviani and Manuel Lama.

We would like to thank the workshop, PhD symposium, and demonstration authors, as well as keynote speakers and workshop Organizing Committees, who together contributed to this important aspect of the conference.

We hope that these proceedings will serve as a valuable reference for researchers and practitioners working in the SOC domain and its emerging applications.

January 2018

Lars Braubach

Juan M. Murillo

Nima Kaviani

Manuel Lama

Loli Burgueño

Naouel Moha

Marc Oriol

ICSOC 2017 Organization

General Chair

Carlos Canal University of Málaga, Spain

Program Chairs

Michael Maximilian	IBM Cloud Labs, USA
Antonio Vallecillo	University of Málaga, Spain
Jianmin Wang	Tsinghua University, China

Steering Committee Liaison

Jian Yang Macquarie University, Australia

Steering Committee

Boualem Benatallah	UNSW, Australia
Fabio Casati	University of Trento, Italy
Bernd J. Krämer	FernUniversität in Hagen, Germany
Winfried Lamersdorf	University of Hamburg, Germany
Heiko Ludwig	IBM, USA
Mike Papazoglou	Tilburg University, The Netherlands
Jian Yang	Macquarie University, Australia
Liang Zhang	Fudan University, China

Workshop Chairs

Lars Braubach	Hochschule Bremen, Germany
Juan M. Murillo	University of Extremadura, Spain

Demonstration Chairs

Nima Kaviani	IBM and Curatio.me, USA
Manuel Lama	University of Santiago de Compostela, Spain

Industry Chairs

Flavio de Paoli	University of Milano-Bicocca, Italy
Antonio Ruiz	University of Seville, Spain

Panel Chairs

Schahram Dustdar Technical University, Vienna, Austria
Michael Sheng University of Adelaide, Australia

PhD Symposium Chairs

Loli Burgueño University of Málaga, Spain
Naouel Moha Université du Québec à Montréal, Canada

Finance Chair

Bernd J. Krämer FernUniversität in Hagen, Germany

Local Organization Chair

Ernesto Pimentel University of Málaga, Spain

Local Organization

Jose M. Álvarez Palomo
Francisco Durán
Nathalie Moreno
Alejandro Pérez Vereda
Mónica Trella

University of Málaga, Spain
University of Málaga, Spain
University of Málaga, Spain
University of Málaga, Spain
University of Málaga, Spain

Publication Chair

Marc Oriol Universitat Politècnica de Catalunya, Spain

Publicity Chairs

Guadalupe Ortiz University of Cádiz, Spain
Juan Manuel Vara Rey Juan Carlos University, Spain
Genoveva Vargas-Solar CNRS, France

Web Chairs

Javier Berrocal
J. Manuel
García-Alonso

University of Extremadura, Spain
University of Extremadura, Spain

Workshop on Adaptive Service-Oriented and Cloud Applications

Mohamed Jmaiel	CRNS, Sfax, Tunisia
Ismael Bouassida	ReDCAD, University of Sfax, Tunisia
Rodriguez	
Khalil Drira	LAAS-CNRS and University of Toulouse, France

**Workshop on IoT Systems Provisioning and Management
for Context-Aware Smart Cities**

Sami Yangui	Concordia University, Montreal, Canada
Mohamed Mohamed	IBM Almaden Research Center, CA, USA
Javier Berrocal	University of Extremadura, Spain
Luca Foschini	University of Bologna, Italy

**Workshop on Engineering Service-Oriented Applications
and Cloud Services**

Andreas S. Andreou	Cyprus University of Technology, Cyprus
Luciano Baresi	Politecnico di Milano, Italy
George Feuerlicht	University of Technology Sydney, Australia
Winfried Lamersdorf	University of Hamburg, Germany
Guadalupe Ortiz	University of Cadiz, Spain
Christian Zirpins	Karlsruhe University of Applied Sciences, Germany

Contents

Adaptive Service-Oriented and Cloud Applications

A BRS Based Approach for Modeling Elastic Cloud Systems.	5
<i>Khaled Khebbab, Hamza Sahli, Nabil Hameurlain, and Faiza Belala</i>	
Detecting Customer Queue “at-risk” Behaviors Based on Ethograms to Minimize Overall Service Dissatisfaction	18
<i>Magali Dubosson, Emmanuel Fragnière, Nathalie Junod, and Bettina Willaerts</i>	
What, Where, When, How and <i>Right</i> of Runtime Adaptation in Service-Oriented Systems	30
<i>Leah Mutanu and Gerald Kotonya</i>	
An End-to-End Security Model for Adaptive Service-Oriented Applications . . .	43
<i>Takoua Abdellatif and Marius Bozga</i>	
Runtime Migration of Applications in a Trans-Cloud Environment	55
<i>Jose Carrasco, Francisco Durán, and Ernesto Pimentel</i>	
Verification of the Consistency of Time-Aware Cyber-Physical Processes . . .	67
<i>Imen Graja, Slim Kallel, Nawal Guermouche, and Ahmed Hadj Kacem</i>	
Model Checking of Cost-Effective Elasticity Strategies in Cloud Computing	80
<i>Rawand Guerfel, Zohra Sbai, and Rahma Ben Ayed</i>	
QoS-Driven Self-adaptation for Critical IoT-Based Systems	93
<i>Arthur Gatouillat, Youakim Badr, and Bertrand Massot</i>	

IoT Systems Provisioning and Management for Context-Aware Smart Cities

BiAgent-Based Model for IoT Applications: Case of a Collision Avoidance System.	111
<i>Souad Marir, Roumeissa Kitouni, Zakaria Benzadri, and Faiza Belala</i>	
Seamless Interactions on the Internet of Things. A Spotify-Based Proof of Concept	124
<i>Jose Garcia-Alonso, Javier Berrocal, Carlos Canal, and Juan M. Murillo</i>	

A Feedback-Based Adaptive Service-Oriented Paradigm for the Internet of Things	137
<i>Yuji Dong, Kaiyu Wan, and Yong Yue</i>	
QoS Prediction for Reliable Service Composition in IoT	149
<i>Gary White, Andrei Palade, and Siobhán Clarke</i>	
Checking and Enforcing Security Through Opacity in Healthcare Applications	161
<i>Rym Zrelli, Moez Yeddes, and Nejib Ben Hadj-Alouane</i>	
Power-Based Device Recognition for Occupancy Detection	174
<i>Azkario Rizky Pratama, Widyawan, Alexander Lazovik, and Marco Aiello</i>	
Cognitive Determination of Policies for Data Management in IoT Systems	188
<i>Aly Megahed, Samir Tata, and Ahmed Nazeem</i>	
A Research Perspective on Fog Computing	198
<i>David Bermbach, Frank Pallas, David García Pérez, Pierluigi Plebani, Maya Anderson, Ronen Kat, and Stefan Tai</i>	
Workshop on Engineering Service-Oriented Applications and Cloud Services	
Lessons Learned from Evaluating Workflow Management Systems	215
<i>Jörg Lenhard, Vincenzo Ferme, Simon Harrer, Matthias Geiger, and Cesare Pautasso</i>	
Sustainable WAsTe Collection (SWAT): One Step Towards Smart and Spotless Cities	228
<i>Daniel J. Rosa-Gallardo, Guadalupe Ortiz, Juan Boubeta-Puig, and Alfonso García-de-Prado</i>	
Designing Suitable Access Control for Web-Connected Smart Home Platforms	240
<i>Sebastian Werner, Frank Pallas, and David Bermbach</i>	
Integrating Smart Devices as Business Process Resources – Concept and Software Prototype	252
<i>Robert Wehlitz, Ingo Rößner, and Bogdan Franczyk</i>	
Architecting Enterprise Applications for the Cloud: The Unicorn Universe Cloud Framework	258
<i>Marek Beranek, Marek Stastny, Vladimir Kovar, and George Feuerlicht</i>	

A Knowledge Carrying Service-Component Architecture for Smart Cyber Physical Systems: An Example Based on Self-documenting Production Systems	270
<i>Christopher Haubeck, Winfried Lamersdorf, and Alexander Fay</i>	
Experiences on Migrating RESTful Web Services to GraphQL	283
<i>Maximilian Vogel, Sebastian Weber, and Christian Zirpins</i>	
Using Risk Patterns to Identify Violations of Data Protection Policies in Cloud Systems	296
<i>Stefan Schoenen, Zoltán Ádám Mann, and Andreas Metzger</i>	
Towards Setting Up a Collaborative Environment to Support Collaborative Business Processes and Services with Social Interactions	308
<i>Andrea Delgado, Laura González, and Daniel Calegari</i>	
Toward an Interactive Mobility Assistant for Multi-modal Transport in Smart Cities	321
<i>Christian Kuster, Nils Masuch, and Fikret Sivrikaya</i>	

PhD Symposium

Meeting IoT Users' Preferences by Emerging Behavior at Run-Time	333
<i>Daniel Flores-Martin</i>	
A Proposition for a Design Method of Service Systems	339
<i>Blagovesta Kostova</i>	
A Model-Driven Approach to Continuous Delivery of Cloud Resources	346
<i>Julio Sandobalin</i>	
SLA-Driven Governance for RESTful Systems	352
<i>Antonio Gamez-Diaz, Pablo Fernandez, and Antonio Ruiz-Cortes</i>	
Towards Adaptive Monitoring Services for Self-Adaptive Software Systems	357
<i>Edith Zavala</i>	
An Approach to Predictive Analysis of Self-Adaptive Systems in Design Time	363
<i>Patrícia Araújo de Oliveira, Francisco Durán, and Ernesto Pimentel</i>	
Paving the Way for a Real-Time Context-Aware Predictive Architecture	369
<i>David Corral-Plaza, Guadalupe Ortiz, and Juan Boubeta-Puig</i>	

Demonstration

OpenTOSCA Injector: Vertical and Horizontal Topology Model Injection	379
<i>Karoline Saatkamp, Uwe Breitenbürger, Kálmán Képes, Frank Leymann, and Michael Zimmermann</i>	
Optimizing Service Delivery with Minimal Runtimes	384
<i>Katharina Gschwind, Constantin Adam, Sastry Duri, Shripad Nadgowda, and Maja Vukovic</i>	
Semantic Data Mediator: Linking Services to Websites	388
<i>Dennis Wolters, Stefan Heindorf, Jonas Kirchhoff, and Gregor Engels</i>	
ARGON: A Tool for Modeling Cloud Resources	393
<i>Julio Sandobalín, Emilio Insfran, and Silvia Abrahão</i>	
Ubiquity: An Extensible Framework for Persistence in Container Environments	398
<i>Mohamed Mohamed, Robert Engel, Amit Warke, and Heiko Ludwig</i>	
Distributed Video Analytics Across Edge and Cloud Using ECHO	402
<i>Aakash Khochare, Pushkara Ravindra, Siva P. Reddy, and Yogesh Simmhan</i>	
Author Index	409

Adaptive Service-Oriented and Cloud Applications

ASOCA: Second Workshop on Adaptive Service-Oriented and Cloud Applications

Mohamed Jmaiel¹, Ismael Bouassida Rodriguez¹, and Khalil Drira²

¹ CRNS, Sfax, Tunisia

² LAAS-CNRS and Univ. de Toulouse, France

{mohamed.jmaiel,bouassida}@redcad.org
khalil@laas.fr

Introduction

The ASOCA 2017 workshop was held in conjunction with the 15th International Conference on Service Oriented Computing (ICSOC 2017) on November 13–16, 2017 in Málaga, Spain. This workshop was focused on the adaptation and reconfiguration issues of the Service-oriented and cloud applications and architectures. It was interesting to see different distributed and centralized approaches on how reconfigurable SOA can repair itself when different execution problems occurs or on monitoring software systems for decision making. Moreover, different technologies were discussed for ensuring non-functional requirements and for the management of different autonomic properties. Some of these works were applied to distributed systems, Cyber-Physical Systems, Internet of Things and Cloud environments. We received 13 submissions, out of which 8 were accepted:

- Verification of the Consistency of Time-aware Cyber-Physical Processes.
- What, Where, When, How and Right of Runtime Adaptation in Service-Oriented Systems.
- A BRS Based Approach for Modeling Elastic Cloud Systems.
- Model checking of cost-effective elasticity strategies in Cloud computing.
- Detecting customer queue “at-risk” behaviors based on ethograms to minimize overall service dissatisfaction.
- An end-to-end security model for Adaptive Service-oriented applications.
- Runtime Migration of Applications in a Trans-Cloud Environment.
- QoS-Driven Self-Adaptation for Critical IoT-Based Systems.

We would like to thank the authors for their submissions, the program committee for their reviewing work, and the organizers of the ICSOC 2017 conference for their support which made this workshop possible.

Workshop Organization

Program Chairs

Mohamed Jmaiel	CRNS, Sfax, Tunisia
Ismael Bouassida	ReDCAD, University of Sfax, Tunisia
Rodriguez	
Khalil Drira	LAAS-CNRS and Univ. de Toulouse, France

Web and Publicity Chair

Emna Feki	ReDCAD, FSEGS, University of Sfax, Tunisia
-----------	--

Program Committee

Abderrahim Ait Wakrime	Télécom SudParis, France
Salah Sadou	IRISA, University of South Brittany, France
Cecilia Rubira	Unicamp, Brazil
Ilia Petrov	TU Darmstadt, Germany
Dimka Karastoyanova	Kuhne Logistics University, Germany
Uwe Zdun	University of Vienna, Austria
Cinzia Cappiello	Politecnico di Milano, Italy
Ian Gorton	SEI, USA
Youakim Badr	INSA-Lyon, France
Amal Gassara	CRNS, Tunisia
Tom Guerout	LAAS-CNRS, France
Djamal Benslimane	Lyon 1 University, France
Mohamed Mosbah	LaBRI - University of Bordeaux, France
Mohamed HadjKacem	University of Sfax, Tunisia
Mouna Rekik	University of Sfax, Tunisia
Ilham Kitouni	Constantine2-Abdelhamid Mehri University, Algeria
Yamine Ait Ameur	IRIT/INPT-ENSEEIHT, France
Claudia Raibulet	University of Milano-Bicocca, Italy
Inen Lahyani	ENIS, Tunisia
Marcos Da Silveira	LIST- Luxembourg Institute of Science and Technology, Luxembourg
Cedric Pruski	LIST- Luxembourg Institute of Science and Technology, Luxembourg
Isabelle Borne	University de Bretagne Sud, France
Flavia Delicato	Federal University of Rio de Janeiro (UFRJ), Brazil
Volker Gruhn	Universitat Duisburg-Essen, Germany

Philippe Roose	LIUPPA, France
Layth Sliman	EFREI, France
Carlos E. Cuesta	Rey Juan Carlos University, Spain
Slim Kallel	ReDCAD, University of Sfax, Tunisia
Ernesto Exposito	Univ. De Pau, France
Sami Yangu	University Concordia, Canada
Samir Tata	Research – Almaden, USA
Samir Medjiah	LAAS-CNRS, France
Michael Mrissa	University De Pau, France
Elisa Yumi Nakagawa	University of Sao Paulo, Brazil
Thais Batista	UFRN, Brazil



A BRS Based Approach for Modeling Elastic Cloud Systems

Khaled Khebbab^{1,2(✉)}, Hamza Sahli¹, Nabil Hameurlain²,
and Faiza Belala¹

¹ LIRE, Constantine 2 University – Abdelhamid Mehri, Constantine, Algeria
{khaled.khebbab, hamza.sahli,

faiza.belala}@univ-constantine2.dz

² LIUPPA, University of Pau, Pau, France

{nabil.hameurlain, khaled.khebbab}@univ-pau.fr

Abstract. Elastic behaviors enable Cloud Systems to auto-adapt to their incoming workloads, by provisioning and releasing computing resources, in a way to ensure a controlled compromise between performance and cost-saving requirements. However, due to the highly fluctuating workloads tendencies, it makes it difficult to predict how a cloud system would behave and to provide precise auto-adaptation action plans. In this paper, we propose a BRS (short for *Biographical Reactive Systems*) based approach to provide a formal description for cloud systems structures and their elastic behaviors using *bigraphs* and *biographical reaction rules*. In addition, elasticity strategies are introduced to encode cloud systems' auto-adaptation policies. Proposed approach is illustrated and evaluated through an example.

Keywords: Cloud computing · Elasticity controller · Strategies
Biographical reactive systems

1 Introduction

Cloud computing is a novel paradigm [16] that has gained a great interest in both industrial and academic sectors. It consists of providing a set of virtualized resources (servers, virtual machines, services, etc.) as on-demand services. These resources are offered by cloud providers according to three fundamental service models: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). The cloud has many characteristics that make it very attractive such as high availability, flexibility and cost effectiveness. However, the most appealing feature for cloud users, and what distinguishes the cloud from the other models, is the elasticity property [23]. In cloud systems [11], elasticity allows to efficiently control resources provisioning according to workload fluctuation, in a way to maintain an adequate quality of service (QoS) while minimizing operating cost. Such behavior is implemented by an elasticity controller, an entity usually based on a closed control loop [22], that decides of the elasticity actions to be triggered (scaling up/down actions [13]).

In the last few years, many academic researches for providing elastic management at different cloud scopes were proposed such as [1, 6, 10, 14, 21]. However, a little

attention has been given to modeling and describing the cloud systems' elasticity controller and its behaviors. In fact, this task can be particularly challenging as these behaviors rely on many overlapping factors such as the available resources, current workload, etc. Managing these dependencies significantly increases the difficulty of modeling cloud systems' elasticity controller. To address this challenge, formal methods characterized by their efficiency, reliability and precision, present an effective solution to deal with all of these aspects.

In this paper, we aim to take a first step towards providing a formal modeling approach that reduces the complexity of designing cloud systems and the elasticity controller behavior. Precisely, we adopt *bigraphical reactive systems* (BRS) [17, 18] as a semantic framework for specifying structural and behavioral aspects of elastic cloud systems. We use bigraphs and bigraphical reaction rules to address both aspects. Reaction rules describe the elastic behavior of a cloud system at service and infrastructure levels according to multiple adaptation rules that are executed by the elasticity controller.

The remainder of the paper is structured as follows. In Sect. 2, we introduce the elasticity controller and identify its components and role in the management of cloud elasticity. We briefly introduce, in Sect. 3, the BRS formalism, apply its modeling approach to specify elastic cloud systems and describe their elastic behaviors by means of adaptation rules. An example of the elasticity management is proposed and evaluated in Sect. 4. In Sect. 5, we review the state of art on formal specification of elastic cloud systems and the use of BRS. Finally, Sect. 6 summarizes the paper and discusses future work.

2 The Elasticity Controller

In elastic cloud systems, resources provisioning can be adjusted by an elasticity controller. It decides the adaptation rules to be triggered, in order to scale the cloud system, in such a way that resource provisioning matches the minimum requirements, as closely as possible. This is done with taking into account many factors as the available resources, current workload, system state, etc. [23]. Structurally, the elasticity controller is usually considered as a closed control loop derived from the IBM's autonomic control loop known as MAPE for *Monitoring, Analyse, Planification and Execution* [22]. In [14, 19], the controller is considered to be constituted by different entities, that interact with each other, to implement the main phases of the control loop. Monitoring and Execution phases are usually considered to be handled by entities that monitor the system (by the means of sensors) and apply actions (using effectors) that the Planification decides, according to flaws that the Analyse identifies.

In our previous work [19], we have provided a cloud systems' design structured in three parts: the *front-end*, the *back-end* and the *elasticity controller*. In this paper, we focus on the elasticity controller and the managed back-end that we see as the cloud hosting environment, i.e., *servers*, *virtual machine* instances (*VMs*) and *service* instances. We abstract the front-end part through the incoming requests which the cloud system receives from clients.

The challenging part here is how to implement a logic that enables the elasticity controller to ensure auto-adaptation behaviors over a managed cloud system. This is accomplished by triggering adaptation rules according to particular conditions, which represent elasticity anomalies to repair. In the next Section, we will adopt a bigraphical approach to model both structural and behavioral aspects of the back-end and the elasticity controller. We introduce elasticity strategies to describe the elasticity controller's behavior by means of bigraphical reaction rules.

3 Formalizing Elastic Cloud Systems with BRS

3.1 BRS Overview

Bigraphical reactive systems (BRS) are a recent formalism introduced by Milner [18], for modeling the temporal and spatial evolution of computation. It provides a graphical model that emphasizes both connectivity and locality. A BRS consists of a set of *bigraphs* and a set of *reaction rules*, which define the dynamic evolution of the system by specifying how the set of bigraphs can be reconfigured.

Graphical Notation and Interface. Figure 1 depicts an example of a bigraph representation. Dashed rectangles denote *regions* describing separate parts of the system. *Nodes* are depicted by circles and represent the physical or logical components of the system. Each node has a type, called *control*, denoted by the labels *A* and *B*. A *signature* is the set of controls of a bigraphs. A node can have zero, one or many *ports* which represent possible connections. Ports are depicted by bullets. In the example, connections are represented as *links*, depicted by curvy lines, which may connect ports and names (*x*, *y* and *z*). They can be considered as (potential) links to other bigraphs. *Sites*, modeled with gray squares, encode parts of the model that have been abstracted away. A bigraph's *B* possibilities to interact with its external environment are visible through its *interface*. For example, $B: 0 \rightarrow < 2, \{x, y\} >$ indicates that *B* has zero sites, two regions and its names are *x* and *y*. Note that a bigraph also has algebraic notations that are equivalent to the graphical ones. For instance, merge product $F | G$ denotes the juxtaposition of bigraphs *F* and *G* which is then placed inside a single region. Nesting operation $F.G$ allows to place bigraph *G* inside *F* and parallel product (\parallel) term may be used to compose bigraphs by juxtaposing their roots and merging their common names. More details about Bigraphs can be found in [17].

Bigraphs Sorting. Classification of controls and links for a bigraph is performed using sorts. A *sorting discipline* is a triple $\Sigma = \{\Theta, K, \Phi\}$, where Θ is a non-empty set of sorts, K is a signature, and Φ is a set of *formation rules*. A formation rule is a set of properties a bigraph has to satisfy. Disjunctive sorts are written as \widehat{ab} , expressing that a node can either be of sort *a* or sort *b*.

Bigraphical Reactive Systems. A Bigraphical Reactive System (BRS) consists of a set of bigraphs representing the state of the system, and a set of reaction rules, defining how the system evolves (by going from one state to another). A reaction rule *Ri* is a pair (R, R') , where *redex* *R* and *reactum* *R'* are bigraphs that have the same interface. The evolution of a system *St* is derived by checking if *R* is a match in *St* (as reference to

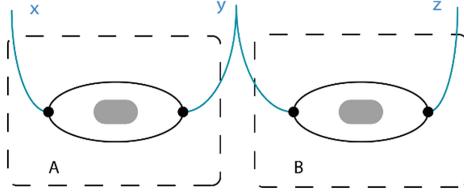


Fig. 1. Example of a bigraph

bigraph matching [4]) and by substituting it with R' to obtain a new system St' . This is made with triggering the suitable reaction rule Ri . The evolution is noted $St \xrightarrow{Ri} St'$.

3.2 A BRS Model for Elastic Cloud Systems

An elastic cloud system is represented by a bigraph CS involving all cloud architectural elements. The bigraph CS is obtained by the parallel composition of the hosting environment (back-end) and the elasticity controller bigraphs (noted B and E) [19]. The sorting logic introduces mapping rules and expresses all the constraints and formation rules, that CS needs to satisfy, to ensure proper and precise encoding of the cloud semantics into BRS concepts. Formal definitions are given in what follows.

Definition 1. Formally, a bigraph CS modeling a elastic cloud system is defined as follows.

$$CS = (V_{CS}, E_{CS}, ctrl_{CS}, CS^P, CS^L) : I_{CS} \rightarrow J_{CS}$$

- V_{CS} and E_{CS} are sets of nodes and edges of the bigraph CS .
- $ctrl_{CS} : V_{CS} \rightarrow K_{CS}$ a control map that assigns each node $v \in V_{CS}$ with a control $k \in K_{CS}$.
- $CS^P = (V_{CS}, ctrl_{CS}, prnt_{CS}) : m_{CS} \rightarrow n_{CS}$ is the place graph of CS where $prnt_{CS} : m_{CS} \uplus V_{CS} \rightarrow V_{CS} \uplus n_{CS}$ is a parent map. m_{CS} and n_{CS} are the number of sites and regions of the bigraph CS .
- $CS^L = (V_{CS}, E_{CS}, ctrl_{CS}, link_{CS}) : X_{CS} \rightarrow Y_{CS}$ represents link graph of CS , where $link_{CS} : X_{CS} \uplus P_{CS} \rightarrow E_{CS} \uplus Y_{CS}$ is a link map, X_{CS} and Y_{CS} are respectively inner and outer names and P_{CS} is the set of ports of CS .
- $I_{CS} = \langle m_{CS}, X_{CS} \rangle$ and $J_{CS} = \langle n_{CS}, Y_{CS} \rangle$ are the inner and outer interfaces of the cloud system bigraph CS .

Definition 2. The sorting discipline associated to CS is a triple $\Sigma_{CS} = \{\Theta_{CS}, K_{CS}, \Phi_{CS}\}$, where Θ_{CS} is a non-empty set of sorts. K_{CS} is its signature, and Φ_{CS} is a set of formation rules associated to the bigraph. Table 1 gives for each cloud concept the mapping rules for BRS equivalence. This consists of the control associated to the entity, its arity (number of ports), its associated sort and graphical notation. *Sorts* are used to distinguish node types for structural purposes and constraints while *controls* identify states and parameters a node can have. For instance, a server noted SE has

control SE^L when it is overloaded and SE^U when unused. A client request, noted q_c , is addressed to a service category identified by the parameter “c”. Also, all nodes representing servers are of sort e and all requests from all clients are of sort q .

Table 2 gives the formation rules Φ_i that bring construction constraints over the BRS specification.

Formation rules give structural constraints over the BRS model. Rule Φ_0 specifies that servers are at the top of the hierarchical order of the deployed entities in back-end bigraph. Rules Φ_{1-3} give the structural disposition of a hosting environment where a server hosts VMs, a VM runs service instances and a service instance handles requests. All connections are port-to-port links to illustrate possible communication capabilities between the different cloud entities. In Φ_{6-7} , we use the name w , for *workload*, to illustrate the connection the cloud system has with its abstracted front-end part. A server is linked to its hosted entities, that represent the resources virtualization (VMs), and a VM is linked to the service instances it is running.

The back-end is managed by the elasticity controller through c-name edges for *control*. The entities are monitored by the *Monitor* node that captures events the *Evaluator* analyses and then triggers actions that the *Effector* applies. Φ_{11} states that the monitor, effector and evaluator entities are always linked. Figure 2 represents a bigraph example which expresses a back-end of a cloud system where two servers (SE), two VMs and two service instances (S) are deployed. V-edges stand for *virtualization* and s-edges for *services*. The elasticity controller bigraph is connected to the back-end with c-name. Squares numbered 1–6 are sites representing parts of the system that are abstracted away.

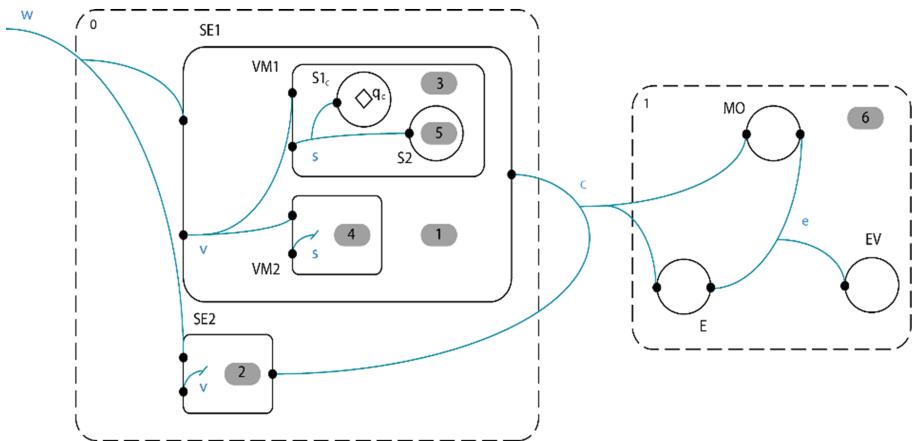
In Rules Φ_{4-5} and 9, active elements may take part in reactions while passive ones won’t. The possible adaptation rules the elasticity controller can take over a cloud system to manage its elasticity will be presented in the next Section.

Table 1. Controls and sorts for the bigraph CS

Cloud element	Control	Arity	Sort	Bigraph	Gr. Notation
Server	SE	3	e	B	Rounded box
Overloaded server	SE^L	2	e	B	Rounded box
Unused server	SE^U	2	e	B	Rounded box
Virtual machine	VM	2	v	B	Rounded box
Overloaded VM	VM^L	2	v	B	Rounded box
Unused VM	VM^U	2	v	B	Rounded box
Service instance	S	1	s	B	Circle
Service instance with a parameter c	S_c	1	s	B	Circle
Overloaded service instance	S^L	1	s	B	Circle
Unused service instance	S^U	1	s	B	Circle
Request	q	0	q	B	Diamond
Request with a parameter c	q_c	0	q	B	Diamond
Evaluator	EV	1	o	E	Circle
Monitor	MO	2	m	E	Circle
Effector	E	2	f	E	Circle

Table 2. Conditions of formation rule Φ_{CS} for the bigraph CS

	Rule description
Φ_0	All children of a 0-region (hosting environment part) have sort e
Φ_1	All children of an e-node have sort v
Φ_2	All children of an v-node have sort s
Φ_3	All children of an s-node have sort q
Φ_4	All \widehat{evs} -nodes are active
Φ_5	All q-nodes are atomic
Φ_6	In a e-node, one port is always linked to a w-name, another port is always linked to a c-name and the other may be linked to v-nodes
Φ_7	In a v-node, one port may be linked to e-nodes and the other may be linked to s-nodes
Φ_8	All children of a 1-region (elasticity controller part) have sort 1, where $1 \in \{o, m, f\}$
Φ_9	All \widehat{om} f-nodes are atomic
Φ_{10}	\widehat{mf} -nodes are always linked to a c-name
Φ_{11}	An o-node is linked to \widehat{mf} -nodes, m-node is linked to \widehat{of} -nodes and f-node is linked to \widehat{of} -nodes

**Fig. 2.** An example of cloud system bigraph CS

3.3 The Elasticity Controller's Behavior

The behavior of the elasticity controller is given as bigraphical reactive rules expressing the dynamicity of an elastic cloud system. In our previous work [19], we defined multiple possible horizontal, vertical, migration and marking actions that the elasticity controller may use to reshape a cloud system at multiple levels.

In this Section, we redefine a set of reaction rules that model the horizontal actions over the cloud hosting environment (servers, VMs and service instances). In addition, we introduce some elasticity strategies that the elasticity controller uses to manage a cloud's elasticity. Table 3 gives the defined actions.

Reaction rules R_i express the possible actions that can be applied over a cloud system. A reaction is triggered when the left-hand side of the rule, that defines a bigraph (*redex*), is a match (or occurs) in CS [4]. The *redex* bigraph is replaced by the right-hand side of the reaction that represents the *reactum* bigraph. Sites nested within different entities (servers, VMs and services) expressed with d , are used to abstract the elements that are not included in the reaction. The specified rules define the horizontal scale elasticity actions, at different cloud levels, for provisioning (R1–3) and de-provisioning (R4–6) resources by scaling-out and scaling-in the hosting environment.

Table 3. Reaction rules modeling elasticity actions in the cloud

Reaction rule	Algebraic form
Duplicate a service instance	$R1 \stackrel{\text{def}}{=} SE.((VM.(S_c.d'') d') d) id$ $\rightarrow SE.((VM.(S_c.d'') (S'_c) d') d) id$
Duplicate a VM instance	$R2 \stackrel{\text{def}}{=} SE.((VM.(S_d'') d') d) id$ $\rightarrow SE.(((VM.(S_d') d') (VM')) d) id$
Turn on a new server	$R3 \stackrel{\text{def}}{=} (SE.d) id \rightarrow (SE.d) (SE') id$
Consolidate a service instance	$R4 \stackrel{\text{def}}{=} SE.((VM.(S_c.d''') (S'_c.d'') d') d) id$ $\rightarrow SE.((VM.(S_c.d'') d') d) id$
Consolidate a VM instance	$R5 \stackrel{\text{def}}{=} SE.(((VM.(S_d''') d'') (VM'.d')) d) id$ $\rightarrow SE.((VM.(S_d'') d') d) id$
Turn off a server	$R6 \stackrel{\text{def}}{=} (SE.d) (SE'.d') id \rightarrow (SE.d) id$

Elasticity Strategies. The defined reaction rules are not sufficient to express the logic that enables the elasticity controller to manage a cloud system’s elasticity. This logic is provided through elasticity strategies that implement the elasticity controller. A strategy describes a behavior to be adopted to manage the elastic behavior in the system. It consists of a set of actions that are triggered in case the specified conditions become true and takes the form [13] *IF condition (s) THEN actions(s)*. In our model, we can encode the strategy conditions by *Bilog* predicates [9], while the actions are modelled by bigraphical reaction rules. For instance, a strategy that executes the condition ($CS \models \varphi$) takes the form below.

$$strat : if CS \models \varphi then R$$

Where φ is a predicate and $B\varphi$ its bigraph encoding where $CS \models \varphi$ is true iff $B\varphi$ is a match in CS . In what follows, we present some strategies with bigraphical representation.

Strategy 1: Hosting Environment Provisioning

When the cloud workload increases by receiving growing number of client requests, the hosting environment needs to scale-out in a way to ensure availability along with performance. Strategy 1 can be expressed with three complementary actions that operate at service and infrastructure level as shown in Table 4.

The predicate φ_1 expresses that service instances of a certain category are overloaded and need to scale-out. The bigraph $B_{\varphi_1} \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}.(S1_{c1}^L.d1)|(S2_{c1}^L.d2)))$ is one possible expression of the predicate for instance. Reaction rule $R1$ is triggered to create a new service instance of the category $c1$. At infrastructure level, when $(CS \models \varphi_2)$ is true, the elasticity controller replicates VM instances by triggering rule $R2$ and new servers are turned online by triggering $R3$, when $(CS \models \varphi_3)$ is true. Also, φ_2 and φ_3 can for example be encoded using bigraphs $B_{\varphi_2} \stackrel{\text{def}}{=} (\text{SE1}^L.(d1)|\text{VM2}^L.(d2))$, respectively $B_{\varphi_3} \stackrel{\text{def}}{=} (\text{SE1}^L.(d1)|\text{SE2}^L.(d2))$.

Table 4. Strategy 1 definition

Level	Condition	Action
Service	(φ_1) Service instances are overloaded	(R1) Replicate service instance
Infrastructure	(φ_2) VMs are overloaded	(R2) Replicate VM instance
	(φ_3) Servers are overloaded	(R3) Turn a new server online

Strategy 2: Hosting Environment De-provisioning

When workload drops, the hosting environment is likely to be over-provisioned and has to scale-in. Elasticity controller performs this behavior at service and infrastructure levels by applying strategy 2 as defined in Table 5.

For instance, if bigraph $B_{\varphi_4} \stackrel{\text{def}}{=} (\text{SE1}.(\text{VM1}.(S1.d1)|(S2^U)))$ is a match in CS, it means that condition $(CS \models \varphi_4)$ is true and the controller triggers reaction rule $R4$. The resulting bigraph would be $B' \stackrel{\text{def}}{=} (\text{SE1}.(S1.d1))$.

Note that more strategies can be specified to perform load-balancing and vertical scale elasticity using the remaining mechanisms defined in [20].

Table 5. Strategy 2 definition

Level	Condition	Action
Service	(φ_4) Service instance is unused	(R4) Consolidate service instance
Infrastructure	(φ_5) VM is unused	(R5) Consolidate VM instance
	(φ_6) Server is unused	(R6) Turn server offline

4 Tool Support

The approach presented in this paper is supported by *MoveElastic*, a framework introduced in [20], for modeling and verifying elastic cloud systems. The framework is based on the integration of the bigraphical cloud system model in Maude, an implementation of *Rewriting Logic* [7, 8]. It enables the specification and execution of elastic behaviors along with the formal verification of elasticity properties.

In the following example, we will present a qualitative evaluation of the two defined strategies through a simple use case.

Example. We present an example of a cloud system to illustrate how the elasticity controller ensures the elastic behavior of a managed cloud system. Consider a voting service running in a private cloud system, with one online server hosting two VM instances and each VM is running five service instances. We consider arbitrary capacity thresholds for each hosting entity (server, VM and service instance). E.g., a server can host up to 4 VM instances, a VM can run up to 10 service instances and a service instance can receive up to 50 client requests at a time. Note that each entity is marked overloaded if its upper threshold is reached and is marked unused when its load reaches zero. Consider an arbitrary workload activity showing a progressive peak of 2500 client requests (phase 1) then slowly drops (phase 2). Graphs shown in Figs. 3, 4 and 5 give the evolution, within 10 time units, of the cloud system as it provisions when the workload rises and de-provisions when it drops. Reconfigurations when the system scales-out/in are actions the elasticity controller triggers when the thresholds are reached.

This scenario shows the *high reactivity* [24] that the elasticity controller provides. We can see that workload variations have no impact over *performance* and that infrastructure *costs* are well controlled. During phase 1, the growing demand is rapidly handled in such a way that the actual capacity slightly overpaces it, this enables eventual further requests to be directly handled before any adaptation is required. Figures 3, 4 and 5 show that the system's processing capacity is never fully achieved and is always half used at the least. Moreover, unnecessarily provisioned resources are rapidly released during phase 2 to prevent from high financial impacts.

From observing the graphs, we can deduce that the elasticity controller operates according to two action plans. The plans $AP1$ and $AP2$ below show the controller's behavior during phase 1 and phase 2 and correspond to strategy 1 and strategy 2. The symbol * indicates zero or more applications of an action or a sequence of actions.

$$AP1 = ((R1* \rightarrow R2)* \rightarrow R3)* \quad AP2 = ((R4* \rightarrow R5)* \rightarrow R6)*$$

It is interesting to see how the cloud system evolves and how elasticity is provided in a cross layered management (infrastructure and service levels). This example is very simple, but it perfectly shows how complex can be the management of even a small cloud system with a single application running with a weak recorded activity comparing to the reality.

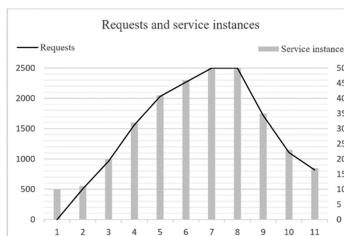


Fig. 3. Service instances provisioning

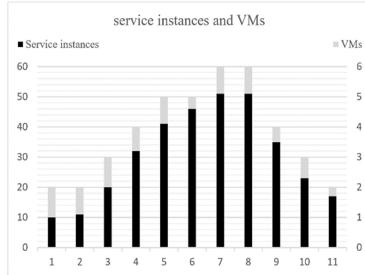


Fig. 4. VMs provisioning

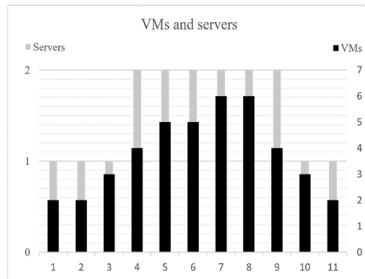


Fig. 5. Servers provisioning

5 Related Work

There have been multiple research studies in the literature using formal methods to specify cloud systems' elastic behaviors. In [3], authors have proposed a formalization based on the CLTLt(D) temporal logic, of several concepts and properties related to the behavior of elastic cloud systems. Qualitative properties of cloud systems have been formally introduced and detailed, such as Elasticity and Resources management, and have been evaluated using an automated verification tool. In this work, precise cloud composition has been abstracted and cloud resources are seen as virtual machines. Authors of [2] have adopted a Petri nets formalization to describe cloud-based business processes' elastic behaviors. Elastic strategies for *routing*, *duplicating* and *consolidating* cloud components at service scope have been provided and compared in terms of reliability and performances. In their work, authors focus on the application layer of a cloud configuration and the infrastructure details are not addressed in the model.

In our previous work [19], we introduced a formal approach based on *bigraphical reactive systems* for modeling both structural and behavioral aspects of elastic cloud systems. Precisely, cloud elastic behaviors are represented in terms of client/application interactions and elasticity methods at distinct levels using *bigraphical reaction rules*.

In this paper, we have focused on the back-end part of a cloud system to specify two strategies that describe the elasticity controller behaviors by means of bigraphical reaction rules. The defined strategies can be combined to provide a cross-layered

elasticity in such a way that the auto-adaptation stability requirements are ensured [25]. This can be made by applying priority levels between and inside the strategies to avoid loops and rules multi-triggering issue. Besides, model-checkers associated to BRS, such as BigMC¹, can be used to verify safety and liveness properties. We consider the BRS formal framework to be suitable in terms of modeling capabilities and we have attempted to illustrate its potential at cloud service and infrastructure levels.

Also, several studies have shown the efficiency and the potential of the BRS Theory to encode and describe complex considerations and behaviors. In [15], authors use BRS as a framework to model Multi Agent Systems. They describe how properties and desiderata for a given BRS specification can be expressed by means of spatial and temporal logics. Precisely, by combining a temporal logic like CTL on top of Bilog, which is a spatial logic for bigraphs [9]. This would enable property-aware matchings and rewritings of bigraphs using tools like BigRED [12] and LibBig². In [5], an extension of BRS called stochastic BRS with sharing (SBRS) is used to model the 802.11 wireless networks protocol. Authors show that conditional reaction rules can be specified to describe very precise matching reactions using *Bilog* predicates. Our goal is to provide a precise property-aware modeling and validation using these solutions.

6 Conclusion

In this paper, we have provided a view of cloud systems including all cloud components that are involved in elastic behaviors. The structural and behavioral aspects of elastic cloud systems have been formalized using the Bigraphical Reactive Systems formalism. Precisely, we use bigraphs to express the structural aspects and bigraphical reactive rules to express the behavioral ones. These behaviors implement the elasticity controller and have been formally expressed by means of two elasticity strategies, for provisioning and de-provisioning cloud systems at service and infrastructure levels. The introduced concepts are illustrated through an example of a managed cloud system where the elastic behaviors are evaluated.

In this present work, we attempt to take a first step towards the formalization of cloud systems elastic behaviors. In the next step, we plan to enlarge our specifications to provide a cross-layered elasticity management of a cloud configuration (at service and infrastructure scopes). Furthermore, we aim to evaluate and experiment our strategies using the *MoveElastic* framework over cloud examples and case studies. Finally, our objective is to provide a complete executable and verifiable formalization of elastic cloud systems.

¹ Bigraphical Model Checker available at <http://bigraph.org/bigmc/>

² A Java library for extensible BRS available at <http://mads.uniud.it/wordpress/downloads/libbig/>

References

1. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: 2012 IEEE Network Operations and Management Symposium, Maui, HI, pp. 204–212 (2012)
2. Amziani, M.: Modeling, evaluation and provisioning of elastic service-based business processes in the cloud. Other [cs.OH]. Institut National des Télécommunications (2015). English. <NNT: 2015TELE0016>
3. Bersani, M., Bianculli, D., Dustdar, S., Gambi, A., Ghezzi, C., Krstić, S.: Towards the formalization of properties of cloudbased elastic systems. In: Proceedings of the 6th International Workshop on Principles of Engineering Service-oriented and Cloud Systems – PESOS 2014, Hyderabad, pp. 38–47 (2014)
4. Birkedal, L., Christoffer Damgaard, T., Glenstrup, A.J., Milner, R.: Matching of Bigraphs. *Electr. Notes Theor. Comput. Sci.* **175**(4), 3–19 (2007)
5. Calder, M., Sevagnani, M.: Modeling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. *Form. Asp. Comput.* **26**(3), 537–561 (2014)
6. Chatziprimou, K., Lano, K., Zschaler, S.: Runtime infrastructure optimisation in cloud IaaS structures. In: CloudCom, vol. 1, pp. 687–692 (2013)
7. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework: How to Specify, Program and Verify Systems in Rewriting Logic. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-71999-1>
8. Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Talcott, C.: Maude Manual, Version 2.6 January 2011
9. Conforti, G., Macedonio, D., Sassone, V.: Spatial logics for bigraphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 766–778. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_62
10. Copil, G., Moldovan, D., Truong, H.-L., Dustdar, S.: Multi-level elasticity control of cloud services. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 429–436. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_31
11. Dustdar, S., Guo, Y., Satzger, B., Truong, H.: Principles of elastic processes. *IEEE Internet Comput.* **15**, 66–71 (2011)
12. Faithfull, A., Perrone, G., Hildebrandt, T.T.: Big red: a development environment for bigraphs. In: Selected Revised Papers from the 4th International Workshop on Graph Computation Models (GCM 2012) (2012). <https://journal.ub.tu-berlin.de/eceasst/article/view/835/829>
13. Galante, G., Bona, L.: A survey on cloud computing elasticity. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing, Chicago, IL, pp. 263–270 (2012)
14. Letondeur, L.: Planification pour la gestion autonome de l'élasticité d'applications dans le cloud. Computer Science [cs], Thesis. Joseph Fourier University (2014). French
15. Mansutti, A., Miculan, M., Peressotti, M.: Multi-agent systems design and prototyping with bigraphical reactive systems. In: Magoutis, K., Pietzuch, P. (eds.) Distributed Applications and Interoperable Systems, DAIS 2014. Lecture Notes in Computer Science, vol. 8460. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43352-2_16
16. Mell, P., Grance, T.: The NIST Definition of Cloud Computing, SP 800–145. National Institute of Standards & Technology, Special Publication (2011)
17. Milner, R.: Bigraphs and their algebra. *Electron. Notes Theor. Comput. Sci.* **209**, 5–19 (2008)

18. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, Cambridge (2009)
19. Sahli, H., Hameurlain, N., Belala, F.: A bigraphical model for specifying elastic cloud systems and their behaviour. *Int. J. Parallel Emergent Distrib. Syst.* (2016). <https://doi.org/10.1080/17445760.2016.1188927>
20. Sahli, H.: Modélisation des Systèmes élastiques Cloud: vers la Vérification Formelle de leur Comportement. Informatique ubiquitaire. Thesis of Constantine 2. Abdelhamid Mehri University (2017). French
21. Trihinas, D., Sofokleous, C., Loulloudes, N., Foudoulis, A., Pallis, G., Dikaiakos, M.D.: Managing and monitoring elastic cloud applications. In: Castelein, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 523–527. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08245-5_42
22. Jacob, B.: A Practical Guide to the IBM Autonomic Computing Toolkit. IBM, International Technical Support Organization, Raleigh (2004)
23. Herbst, N., Kounev, S., Reussner, R.: Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the 10th International Conference on Autonomic Computing. USENIX, San Jose (2013)
24. Dupont, S., Lejeune, J., Alvares, F., Ledoux, T.: Experimental analysis on autonomic strategies for cloud elasticity. In: Proceedings of 2015 IEEE International Conference on Cloud and Autonomic Computing (ICCAC), Cambridge, USA, pp. 89–90, September 2015
25. Sama, M., Elbaum, S.G., Raimondi, F., Rosenblum, D.S., Wang, Z.: Context-aware adaptive applications: fault patterns and their automated identification. *IEEE Trans. Softw. Eng.* **36** (5), 644–661 (2010)



Detecting Customer Queue “at-risk” Behaviors Based on Ethograms to Minimize Overall Service Dissatisfaction

Magali Dubosson¹, Emmanuel Fragnière^{1,2(✉)}, Nathalie Junod¹,
and Bettina Willaerts¹

¹ HES-SO, Service Design Lab,

Le Foyer - Techno-Pôle 1, 3960 Sierre, Switzerland

magali.dubosson@hefr.ch, emmanuel.fragniere@hevs.ch,

nathalie.junod@hesge.ch, bwillaerts@gmail.com

² University of Bath, Bath BA2 7AY, UK

Abstract. Every service encounter corresponds to a “queue network” in which a system of waiting lines is connected to servers. We posit that each production service type (e.g., restaurant, airport) requires an adapted queue design in order to maximize attributes salient to customers (i.e., their primary elements of perceived value) in today’s globalized service environment. While the queues have been studied from many angles, a scientific contribution based on a human ethology approach proposing the early identification of “at-risk” behaviors to regulate queue dynamics seems to be novel. To remedy this shortcoming, the large-scale food distribution sector has been chosen for the application of a naturalistic observation approach to describe in detail the behavior of customer queues. Sixteen immersion episodes were conducted in the months between May and June 2016. Using RQDA, we analyzed the immersion transcripts and identified typical customer queue behavioral patterns. Then, we developed an ethogram containing what we considered to be “at-risk” queue behaviors. This ethogram can ultimately be used as an anticipatory indicator in the context of feedforward management controls. Feedforward control, as opposed to classical feedback controls, is based on the early detection of risks and the implementation of mitigation before damage occurs. While this approach requires human attention and expertise (which can be labor-intensive), there is also potential for human ethology to assist managers with supportive or complementary automation. Indeed, the factual description of behaviors contained in our ethogram can easily be coded with modern technology like facial expression and body recognition technologies.

Keywords: Enterprise risk management · Human ethology · Service science · Waiting line management

1 Context

For centuries humankind has shown no lack of imagination toward reducing distances; in particular, with the development cars, planes and even rockets. It must be observed, however, that queues still exist and are part of our daily lives. At the time of the instantaneousness of the Internet, the rule of “Anytime, Anywhere, Any device” takes on its full meaning, and customers seem less ready to wait. Indeed, according to a study conducted by IFOP (www.ifop.com - based on a sample of 1000 people aged 18 and over), no less than 30% of French consumers would abandon a purchase just because of a checkout queue. Consequently, “balking” or “reneging” behaviors and feelings of frustration and injustice have an impact in terms of costs for the company and can create a negative perception of the service offered to the client. The design of suitable queues therefore seems to be an element to be considered. In this paper we see how human ethology precepts can contribute to better queue management. Various queuing systems adopting the FIFO (First In First Out) logic have emerged alongside the traditional queue, which is known as the multiple-queue configuration. The serpentine queue also warrants mention. It is a single queue leading at the end to multiple servers. Another important type of file configuration, the queue with ticket-number, is also known as the virtual queue configuration. A first line leads to a ticket dispenser. Then, each client waits until his/her turn. More recently, the virtual ticket number has gained popularity thanks to the Internet where physical presence is not required anymore. This research aims to manage the annoyances associated with customer queues through human ethology methodologies. The queuing behaviors identified during immersions are collected in an ethogram, by sensorial qualification. On the basis of naturalistic observation, queue ethograms are established. The goal is then to be able, in real time, to prevent risks of overflow, general dissatisfaction, and also generally to promote a better perception of value offered by the final service. The entirety of the fieldwork, based on participative observations, was carried out according to the methods of human ethology in the food retail sector. A first “exploratory” field phase, including four immersions, allowed the observer to become familiar with the dynamics of the points of sale. In addition to a brief observation of the queues, the customer’s purchasing context was also examined. The second “in-depth” field phase, consisting of sixteen immersion episodes (10 min each), enabled us to identify typical behaviors of clients in the queues. This inventory of behaviors was made possible thanks to the development of an observation grid and the processing of its data with the RQDA program. Thereafter, an ethogram was created to highlight certain behaviors identified as potentially “at risk”. This paper is organized as follows. In Sect. 2, we propose a simple explanation about of queuing theory, as well as human ethology adapted to this context. In Sect. 3, we propose a brief literature review of queuing theory and related developments in the psychology of waiting lines. We also explain the main principles of human ethology in the context of customer queues. In Sect. 4, we present the methodology employed to gather data through naturalistic observation. In Sect. 5 we propose an ethogram of at-risk queue behavior based on fieldwork analysis. We conclude by indicating research limitations as well as further research directions.

2 Queuing and Human Ethology

It was in the beginning of the 20th century that the Danish engineer Agner Krarup Erlang (1878–1929) developed the mathematical theory of waiting lines (called queuing theory). His mathematical models were applied for the first time to assessing the very long queues of white-collar workers who wanted to take skyscraper lifts to their offices. Erlang was not only interested in quantitative models to evaluate waiting time and queue length, but also in the perception of time spent in the queue (qualitative aspects, such as talking with a colleague, reading a newspaper or enjoying a bagel and a coffee, may create the impression that the perceived wait time is shorter than the actual wait time). Unfortunately, today, the two approaches (quantitative and qualitative) are too often treated separately. On the one hand, quantitative approaches coupled with Monte-Carlo simulation techniques now allow computers to calculate a range of indicators to manage the queue from an essentially operational point of view. For example, when wait times in the queue are considered too long, these models can accurately assess the number of additional cashiers needed in order to return to a reasonable wait time. On the other hand, qualitative approaches allow us to grasp behavioral aspects of the queue. The most studied behaviors are: *a priori* impatience (“balking”) and *a posteriori* (“reneging”), which lead to the abandonment of the queue if it is perceived as too long, as well as the passage from one queue to another (“Jockeying”) in order to reduce the waiting time. For example, jockeying is a typical behavior of the traditional queue and typically creates feelings of injustice within the queue. The serpentine queue eliminates the possibility of “jockeying” on the other hand. However, it is subject to more frequent abandonment behaviors when the queues are very long. The same applies to the queue with ticket-number. First, we consider the traditional queue that consists of several queues each leading to a counter. Next, we consider the single line leading to several cashiers, also called serpentine, which consists of a single queue leading to several cashiers. The last queue configuration we consider is the queue with ticket, also called the virtual queue. A first line leads to the ticket dispenser. Then, people take their places in the waiting room and go to a specific window as soon as the electronic bulletin board invites them. All three queue configurations apply the so-called FIFO (First In First Out) rule. The term “human ethology”, is defined as the study of human behavior without relying on questions or discussion but through sole observation from the outside. Eibl-Eibesfeldt [1] is known to be the first scientist to have systematically explored human ecosystems with methods and concepts of animal ethology. In the 1970s, he created a repertoire of the most universal human behaviors. Basic facial and gestural expressions, appearing to be found in all human societies, have thus been categorized. He also highlighted the invariance of certain behaviors such as frowns related to the expression of anger or in opposition, eyebrow shrugging (frowning), as a sign of friendly welcome. The energy leading to a given behavior is put in place by two different triggering systems that can be seen as the dual causes of behaviors:

- Behavior of endogenous origin (internal stimulations). This is defined very well through the notion of drive, referring to everything that is of physiological origin.

- Behavior of exogenous origin (external stimulations). The exogenous origin concerns the environment and includes the spatio-temporal dimensions proper to it, but also the presence and activities of other individuals or groups of individuals.

These fundamental aspects are part of a “stimulus-response” pattern of cause and effect. Still, in order to understand the notion of behavior, a theoretical contribution can be made with regard to innately acquired behaviors. In the majority of cases, these are closely intertwined in the behaviors observed. Indeed, these coexist in our brain and are dictated by two parts of the brain. In the case of innate behaviors, they are located in the paleo-cortex (reptilian brain), while the acquired behaviors find their seat in the neo-cortex. We can thus define two main types of behaviors:

- Inborn Behaviors. Innate behaviors are dependent on the hereditary heritage of the species and are registered in the genes. These are all instinctive behaviors. Among these are the expression of emotions, a baby’s instinctive behaviors, walking, and basic social recognition or communication behaviors. These are dictated by the reptilian brain.
- Acquired Behavior. The acquired behaviors result from various learning and experiences acquired during the ontogenesis of the individual.

Take the example of escalators in London. It is known that users are placed on the right side of the escalator one after the other so that the people pressed for time can take the left side. This behavior has been learned and is not innate. Let us take another example of queuing behavior that is no longer learned, but is innate. A given queue system is configured as a traditional queue (multiple queues in parallel). We know that as 90% of people are right-handed, the right queues have a natural tendency to fill more strongly than those on the left. It is as if most of us are driven by our right hand (this is the case when we shop). Now that these basic notions directly related to behavior have been developed, we can clarify what we mean by:

- Typical behavior of queues. One of the objectives of this work is the observation of typical behaviors of queues. By this we mean observing typical behavioral patterns that occur repeatedly in queues during field observations. [2] provides an example of a naturalistic observation approach employed to understand skiers’ behaviors in cable cars queues in order to improve their overall satisfaction.
- At-risk queue behaviors. Once observing the set of standard behaviors of queues, one of the tasks will be to highlight the at-risk behaviors. By the notion of risk behaviors we mean behaviors that may have a negative impact on the issuer of the behavior or on its direct or indirect environment and which can also impact the image of the service provider.

3 Literature Review

Below is an overview of research that takes into account some behavioral aspects of queues. Maister [3] was one of the first authors to discuss the psychology of queues. [4] emphasized the importance of the control process and the announcement of wait times

in order to maintain a sense of justice in the queue. [4] proposes a variety of stress reduction mechanisms, such as providing clients with a wait-time forecast, or offering clients fast-pass options. These practical suggestions can help managers reduce perceived waiting time, improve the customer's waiting experience, and generally improve the management of the queue. [5] demonstrates that some elements, such as music or queue structure, if properly managed, can have a positive impact on perception of service expectation and satisfaction. The paper concludes with a comprehensive model, including all the elements that the authors judge to affect overall perceptions of service expectations. [6] shows that music or perfume can reduce the level of discomfort during a wait. The overall satisfaction level of the service therefore increases with the insertion of external stimuli during a queue. [7] highlights the importance of the relationship between the structure of the queue and the attitudes of clients. The authors suggest that individuals who wait in a serpentine queue system are more awakened than those waiting in a multiple queue system (i.e. traditional).

The authors also question the perceived anxiety about whether the service will be delivered according to expectations. [8] developed an econometric model to explain wait in services. The authors show that some independent variables such as the human factor and visual elements have a significant influence on the perception of clients' expectations. [9] describes waiting as a psychological experience. The authors of this paper find that the traditional queue can produce a sense of injustice, even if from an objective point of view there is no inequality. The authors showed that when the client perceives the service provider has control over the wait time, longer wait times will be more unpleasant. [10] examines the difference between actual wait times and wait times as perceived by the consumer, based on a case study. It becomes clear that there are different ways to reduce this difference, and that depending on the emotional state of clients, the perceived wait time may be longer or shorter. It is important to note that with the advent of new technologies and the Internet, qualitative research has also focused on the perception of waiting for online services with, for example. [11] studied the wait time tolerated when consulting websites. [12] highlighted through their study that a service involving a feeling of discrimination can lead to frustration and a sense of helplessness to the client. Finally, Hall [13], who developed proxemics, used to say that individuals tend to create an emotionally strong zone around themselves, which may also be described as an individual perimeter of security, like a bubble.

4 Methodology

There are two types of approaches in ethology to collect data: naturalistic observation and experimental manipulation. The naturalistic observation is made in a natural environment or in a reconstitution of it. Experimental manipulation is carried out in the laboratory and attempts to control all induced variables. Here, we solely employ naturalistic observation as an exploratory approach to discover behavioral patterns related to customer queues. The working tools used in naturalistic observation are primarily paper/pencil, but also photos, videos or even voice recordings. The basics of inquiry are always based on the following three precepts: to remain at a descriptive level, to determine what is observed and to develop an ethogram (catalog of behaviors in a

certain context). The research was carried out with customers attending supermarkets. The framework for large-scale food retailing was chosen for several reasons. Indeed, this sector is represented by large players daily engaged in a major competitive battle. Offering essential goods, these places have become essential and have to cope regularly with the large flow of customers. The clients waiting in line were observed by a unique person (one of the authors of this paper) according to the methods of human ethology. The idea of an ethological approach consists in a direct observation, with an outside and non-interactive observer, of the different behaviors encountered within the human species, here of the clients. Ethology, whose general characteristics have been developed in the previous section, seems to be a relevant and meaningful discipline for the researcher who wishes to understand the ecosystem of queues with the intention of describing related behaviors in a precise manner. The objectives mentioned in the preceding point can be achieved by the following three human ethological precepts: staying at a descriptive level; determining what is observed; developing an Ethogram. The subjects observed are the customers making their purchases in the selected outlets and who take part in a queue to pay a cashier for what they have bought. The observer is part of the group of customers waiting in line so as to be able to collect precise information concerning the queues and ensure that the observer is not encumbered by a visual obstacle. It is a participatory approach. The proximity to the group allows for better listening to the enveloping noise. Typically, the quality of the observations gathered depends on the mode of recording. Indeed, if the use of a video camera makes it possible to view the events later and to have a very fine screening, this technique offers only a two-dimensional view of the events and often requires additional lighting. Moreover, this type of equipment can have a disruptive role, especially with humans. A less expensive and simpler technique is used in this research: paper and pencil. It is well accepted in the field of ethology that paper and pencil are a relevant approach to make live observations and remain quite valid without requiring the intervention of more complex tools [14]. So, in sum, here are the main characteristics of the fieldwork undertaken for this study. We worked in two different locations (a small store and a large store). In both cases, queue configurations corresponded to the traditional line system; that is, multiple queues. The sampling technique adopted was “centration successive” (focal sampling): observation began as soon as the client entered the queue and continued until the client left the queue [15]. Sixteen immersion episodes (eight per store) were conducted, at 10 min each, on four Saturday afternoons (known to be four busy days). An observation grid containing pre-identified categories of queue behaviors was developed during a preliminary phase conducted in several commercial centers before the 16 formal immersion episodes. The transcripts of the 16 immersions episodes were encoded in the RQDA software (RQDA - R package for computer assisted qualitative data analysis) that allowed the grouping of similar data by coding the data in categories. This was used to produce tables containing typically observed queue behaviors.

In this case, there the two supermarkets observed will remain anonymous. This type of approach allows an in-depth understanding of the behaviors of individuals in relation to the complexity of the environment. Practically, the overall research design can be summarized in the three following phases:

Phase 1: Pre-immersion. The first step consists of a phase of exploratory observation or reconnaissance observation carried out according to unstructured sampling techniques. In this case, the most important and interesting elements were observed in the supermarket. This first exploratory stage allows for control of the environment studied, but also a familiarization of the observer with the main lines of behavior of the clients in the context of a queue, which allows a first “slimming”. At this stage, it is the purchasing context of the customer that is taken into account and not just the queue item leading to a cash register. The advantage of this approach is not to carry out a reductive analysis limited only to the observation of the individual when he/she is in the queue, but rather to adopt a more global approach integrating the aspects related to the context in which the client has previously evolved.

Phase 2: In-depth Phase. The second phase consists of field observation work focusing on the behaviors present in the queues. This phase is possibly based on certain points highlighted during the previous phase, the exploratory phase. With regard to observations in the checkout queues, the client is considered observable for the typical behavior of the queues from the moment that the client goes to a queue until he/she leaves the cashier after collecting and paying for his/her purchases. Once these sixteen immersion episodes were completed, accurately collected and described observations were encoded into the computer for processing. The encoding and analysis of the data from the observation grids was carried out from the data processing system “RQDA-R package for computer assisted qualitative data analysis.” Indeed, this software allows for the regrouping of similar data thanks to the coding of the data.

Phase 3: Realization of the Ethogram. Before developing this third step, it was first necessary to develop the term ethogram in order to have a complete understanding of the chosen methodology. Let us first recall that the ethological method differs from other methodological approaches, in particular by the importance it places upon the descriptive phase of the observation. This step involves the development of an ethogram that can also be seen as a repertory comprising the characteristic behaviors of an individual within a specific framework. In this inventory, behaviors are classified or deconstructed according to previously established behavioral categories. The implementation of an ethogram involving various risk behaviors allows for the implementation of appropriate managerial recommendations.

5 Findings

The analysis of the immersion transcripts has provided many detailed insights regarding queue behaviors. In this section, we present a summary of this analysis. Then, we illustrate the method by which we developed an ethogram of “at-risk” queue behaviors based on this analysis, which is in turn intended for decision-making purposes. There are several scenarios for arriving in a traditional queue. Some customers go directly to the cashier closest to them as soon as they leave the department, even if it is not the fastest queue. The majority of this type of customer is often distracted by a ringing mobile phone, a telephone conversation, listening to music, or sending messages. Other customers mark a brief stop with a head swing from left to right in order to

choose the queue they want to take part in (more marked in the large store). In the majority of cases, the client will take part in the queue that includes the fewest clients or the queue that is shortest in length. Some customers make detours without pause and head to a particular check-out counter. There are also customers who simply follow the customer who precedes them. The fluidity of the queue decreases during periods of high attendance at the check-out counter. In the off-peak period, too many check-out counters are open. The opening of an extra check-out counter is not always clearly communicated, and some customers are not even aware of it. Indeed, the green lights above the check-out counters are not very visible, and some cashiers do not systematically offer oral invitation to the customer to go to a newly opened queue when this is the case. Regarding specific observations conducted in the small store, arrival in the queue was sometimes more complex. The client did not always understand whether it was a single or multiple queues leading to several check-out counters. The queues took on the movement of an accordion, and their “rhythm” was dependent on the different periods of business activity and client behavior. Indeed, it has been often observed that the client moved from a “rest” state to an “active” state within the same queue. The customer took action to deposit his/her goods on the carpet, on receipt of goods as soon as they had been scanned, and also when paying for his/her purchases. The customer thus went through several implicit stages and was solicited periodically. Various behaviors can be observed at different stages of the same queue and for the same client. Most of the time, the client seems lonely, but some social behaviors may also appear. Various external events animated the ecosystem of the queue positively as well as negatively. No major incidents were noted. The customer often seemed to be “watching his/her back,” looking at the moves of his/her eyes to the left and to the right. This was especially the case when a certain safety zone (in terms of distance between people) was not respected. Some clients also prepared their wallet, payment card, vouchers, etc., in advance. The shape of the traditional queue was generally a straight line. However, when an unforeseen element had to be taken into account, it became immediately more disordered. Signaling was good but did not easily allow the customer to leave the queue for one reason or the other (being regularly cluttered with shopping carts or shopping baskets of customers and thus making it difficult to maneuver). The client then had to clear a path to leave the queue. In Table 1, we present a formalized way of describing queue behavioral patterns based on our analysis along with frequencies (i.e., subjective probabilities). Then, based on formalized tables such as Table 1, we developed an ethogram (see Table 2) containing queue behaviors that we consider to be at-risk behaviors in terms of the overall service value perception. There are several forms of ethograms. However, most include a first column with the name of a typical behavior, followed by a column describing this behavior more accurately.

The value of our approach is really related to the logic of a preventive control (see below) in which mitigation triggers are based on the ethogram. The last methodological element used in this research project was preventive control (feedforward control) [16] for addressing human risks [17]. Preventive control allows the indicators coming from the ethogram to trigger the “mitigations” necessary to reduce emotional risks in the queue. According to the IIA (Institute of Internal Auditors, www.theiia.org), control means any action taken by management, the board or other parties to enhance risk

Table 1. Extract of inventoried behaviors ranked according to their frequency.

Category	Description of observed behaviors	Frequency
<i>Reneging</i>	The customer informs the cashier of being in a hurry and leaves the line without completing his/her purchases. The client leaves the queue because it does not advance fast enough and observes the other queues.	Rare
<i>Overtaking</i>	The customer leaves the line without informing the customers he/she plans to return later, then passes in front of the other customers.	Rare
<i>Surveillance</i>	The client seems to open his eyes and stretch his head to the left to assess the progress of the queue. The client keeps his/her bust straight. Customer turns head from left to right. The customer observes the maneuvers of the previous customer.	Likely

Table 2. An ethogram describing “at-risk” queue behaviors.

Category	Behavior	Description of expected behaviors
<i>Distraction</i>	Looking around	The client observes how the queue is moving. The client stares at the service provider.
<i>Boredom</i>	Getting impatient	The client glances at his watch repeatedly. The client taps fingers on the conveyor belt. The client taps foot on the ground.
	Switching lines	The client leaves his queue and takes part in another queue.
<i>Dissatisfaction</i>	Giving up	The client abandons the queue and leaves the store.
<i>Frustration</i>	Assaulting	The client verbally assaults another client because the latter misunderstands the signaling of the queue.
	Avoiding	The client returns to the end of the queue without complaining.
<i>Nervousness</i>	Getting upset	Under queue pressure, the client frowns.
	Fussing	The client scratches his head. The client plays with his car keys. The client actively searches in his bag.

management and increase the likelihood that established objectives and goals will be achieved. Control corresponds to an essential part of management and is comprised of four critical components: planning, organization, employee involvement and control (assuring the first three components). Risk control is a three-step process: 1. Definition of the objective; 2. Measuring the achievement of the target within the time limit; 3. The managerial corrective phase (do nothing if goal is reached, reconsider objective if too ambitious, set up risk mitigation treatments to increase the likelihood of achieving the objective). When control is corrective (i.e. feedback control), the managerial corrective phase is triggered on the basis of an objective measurement of the achievement

of the result; hence afterwards. When the control is anticipated (i.e. feedforward control), the actual measurement of the achievement of the result is replaced by a forecast of it. The actions of the corrective phase can therefore be taken more quickly, even before the deadline for achieving the target has been reached. A feedforward control can be presented as a process in three steps, as in the following example implementing the ethogram:

- (1) Determining an objective related to the customer queue (e.g., maintaining a “relaxed” queue during rush hours);
- (2) Analyzing indicators from the ethogram (e.g., diagnosis of too much nervousness in the queue);
- (3) Triggering “mitigation” according to the level of the indicators (e.g., distributing glasses of water and orange juice in order to lower the “emotional temperature” and to get back to normal state).

Risk controls are easily programmed, thanks to their linear structure. However, the feedforward control compared to the feedback control is trickier since the anticipative indicators usually come from an expert. In our case, it would come from the ethogram, so it could be automated.

6 Conclusion and Further Research

The purpose of this research is to identify early signals to avoid a sudden increase of emotional temperature related to recurrent annoying experiences when waiting in line. The originality of this work is that at-risk queue behaviors were studied through human ethology. Ethology is the study of the behavior of living things (animal or human) in their “natural ecology”. This discipline focuses on all of the factors that induce certain behaviors (stimuli, innate, learned...). Through naturalistic observation, we attempted to identify the typical behavior of customer queues: it was accomplished through observation of typical behaviors of queues occurring several times during our field observations. In view of this, the objectives set out in this paper were to observe different queues and to inventory the various behaviors, in order to be able to detect risk behaviors typically associated with waiting lines. Thanks to this early detection of at-risk behaviors, managers are able to implement adapted solutions to return the queue to a normal state. The fieldwork took place as follows: identification of the different types of queues, identification of analyzed behaviors, and grouping of these in an ethogram by sensorial qualification. Once these steps were completed, various emotional and physical mitigation tools were proposed. While we discovered that some clients were easily distracted, our observations also pointed out that not only can impatient behavior have a contagious effect on the rest of the queue, it may also lead to other risk behaviors such as change of queue or abandonment. Faced with these results, various managerial recommendations have been put in place to improve the management of queues. These include taking into account the context in which the customer makes his purchases, as well as the design of a single, personalized queue that meets the needs of the company. The organization of training seminars in human ethology to better understand customer behavior [18] also complements the proposed solutions, in

order to better know and manage the problematic object of this study. Finally, we show how preventive controls (feedforward controls) based on at-risk queue behavior ethograms can be easily implemented to practically mitigate the typical physical and emotional responses observed in the queues in order to avoid too sudden a climb in “emotional temperature”. The main limitations of this work are that naturalistic observation corresponds to exploratory research and that the ethogram produced is not validated. So, our findings cannot be generalized. Nevertheless, this approach has enabled us to discover things that we are personally experiencing every day, and that if well addressed in a practical manner could improve the lives of customers. We intend in the future to study more collective behaviors, and not just individual behaviors observed in the queue, such as mimicry, contagion or crowd effects. Moreover, it would also be interesting to study the possible occurrence of new behaviors following the massive introduction of digitization in many outlets. This work could be complemented by a comparison between digital and traditional. We also intend to develop a research collaboration with an institute specializing in facial expression and body recognition technologies in order to automate our approach.

References

1. Eibl-Eiblfsfeldt, I.: *Human Ethology*. Aldine de Gruyter, New York (1999)
2. Fragnière, E., Gaillet, V., Nanchen, B., Ramseyer, R.: Using ethological approaches to understand skiers' behavior in cable cars queues in order to improve overall satisfaction: an empirical study conducted in the Swiss Alps. In: Za, S., Drăgoicea, M. (eds.) IESS 2017. LNBIP, vol. 279, pp. 421–430. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56925-3_33
3. Maister, D.: The psychology of waiting lines. In: Czepiel, J.A., Solomon, M.R. (eds.) *The Service Encounter: Managing Employee/Customer Interaction in Service Businesses*. D. C. Heath and Company, Lexington Books, Lexington (1985)
4. Naumann, S., Miles, J.: Managing waiting patients' perceptions: the role of process control. *J. Manag. Med.* **15**(5), 376–386 (2001)
5. Baker, J., Cameron, M.: The effects of the service environment on affect and consumer perception of waiting time: an integrative review and research propositions. *J. Acad. Mark. Sci.* **24**(4), 338–349 (1996)
6. McDonnell, J.: Music, scent and time preferences for waiting lines. *Int. J. Bank Mark.* **25**(4), 223–237 (2007)
7. Rafaeli, A., Barron, G., Haber, K.: The effects of queue structure on attitudes. *J. Serv. Res.* **5**(2), 125–139 (2002)
8. Choongbeom, C., Atul, S.: Assessing the relationship between waiting services and customer satisfaction in family restaurants. *J. Qual. Assur. Hosp. Tour.* **13**(1), 24–36 (2012)
9. Van Riel, A.C.R., Semeijn, J., Ribbink, D., Bomert-Peters, Y.: Waiting for service at the checkout: negative emotional responses, store image and overall satisfaction. *J. Serv. Manag.* **23**(2), 144–169 (2012)
10. Whiting, A., Donthu, N.: Closing the gap between perceived and actual waiting times in a call center: results from a field study. *J. Serv. Mark.* **23**(5), 279–288 (2009)
11. Nah, F.: A study on tolerable waiting time: how long are web users willing to wait? *Behav. Inf. Tech.* **23**(3), 153–163 (2004)

12. Klinner, N.S., Walsh, G.: Customer perceptions of discrimination in service deliveries: construction and validation of a measurement instrument. *J. Bus. Res.* **66**(5), 651–658 (2013)
13. Hall, E.T.: *The Hidden Dimension*. Anchor Books, New York (1966)
14. Lehner, P.N.: *Handbook of Ethological Methods*, 2nd edn. Cambridge University, New York (1996)
15. Altmann, J.: Observational study of behavior: sampling method. *Behaviour* **49**, 227–265 (1974)
16. Fragnière, E., Sullivan, G.: *Risk Management: Safeguarding Company Assets*. Crisp, USA (2007)
17. Fragnière, E., Junod, N.: The emergent evolution of human risks in service companies due to control industrialization: an empirical research. *J. Financ. Transform.* **30**, 169–177 (2010)
18. Vargo, S.L., Maglio, P.P., Akaka, M.A.: On value and value co-creation: a service systems and service logic perspective. *Eur. Manag. J.* **26**(5), 145–152 (2008)



What, Where, When, How and *Right* of Runtime Adaptation in Service-Oriented Systems

Leah Mutanu¹⁽⁾ and Gerald Kotonya²

¹ School of Science and Technology, United States International University,
Nairobi, Kenya

lmutanu@usiu.ac.ke

² School of Computing and Communications,
Lancaster University, Lancaster, UK
gerald@comp.lancs.ac.uk

Abstract. Software as a Service reflects a “service-oriented” approach to software development that is based on the notion of composing applications by discovering and invoking network-available services to accomplish some task. However, as more business organizations adopt service-oriented solutions and the demands on them grow, the problem of ensuring that the software systems can adapt fast and effectively to changing business needs, changes in their runtime environment and failures in provided services has become an increasingly important research problem. Dynamic adaptation has been proposed as a way to address the problem. However, for adaptation to be effective several other factors need to be considered. This paper identifies the key factors that influence runtime adaptation in service-oriented systems, and examines how well they are addressed in 29 adaptation approaches intended to support service-oriented systems.

Keywords: Service-oriented · Runtime adaptation · Validation

1 Introduction

Service-Oriented Architecture (SOA) provides the conceptual framework for realizing service-oriented systems (SOS’s) by supporting dynamic composition and reconfiguration of software systems from networked software services [1]. Rosen [2] identifies the key motivations for SOA as agility, flexibility, reuse, integration and reduced cost. However, the need to ensure that the systems can adapt quickly and effectively to changing business needs, changes in system quality and changes in their runtime environment is an increasingly important research problem [3]. Effective adaptation ensures the system remains relevant in a changing environment and is an accurate reflection of user expectations.

Taylor [4] defines dynamic adaptation as the ability of a software system’s functionality to be changed at runtime without requiring a system reload or restart. Taylor points out that there is an increasing demand for non-stop systems, as well as a desire to avoid annoying users. However, current approaches for supporting runtime adaptation

in service-oriented systems differ widely with respect to the nature of systems they support, the types of system changes they support and their underlying model of adaptation [5, 6]. In addition, it is also unclear how these approaches address the important issue of ensuring the adaptation is effective. A growing consensus amongst researchers is that runtime adaptation in SOA should incorporate a validation element [16, 18].

In their research roadmap for self-adaptive systems, Lemos et al. [7] emphasize the need for feedback control in the life cycle of self-adaptive systems, and the need to perform traditional design-time verification and validation at runtime. In another survey, Salehie et al. [45] note that testing and assurance are probably the least focused phases in the engineering of self-adaptive software. Papazoglou et al. [18] echo this view. They note that the bulk of research in adaptive service-oriented systems has focused largely on dynamic compositions. Adaptation validation goes beyond verifying that the adaptation conforms to its operational specification. Validation is concerned with verifying the acceptability of an adaptation, often from the point of view of the system user – i.e. is it the right adaptation for the problem as opposed to whether it is specified right? Validation assesses the effectiveness of an adaptation. Because user requirements are constantly changing, a self-validation process would enable the adaptation system to self-assess and self-evolve in order to remain relevant.

This paper identifies the key factors that influence adaptation in service-oriented systems, and uses them to review 29 approaches intended to support runtime adaptation in service-oriented systems. The survey presented in this paper compliments and extends existing surveys on adaptation by examining runtime adaptation from a different, but useful perspective and extending the review to incorporate validation as key factor. The paper is organised as follows, Sect. 2 identifies the factors that influence adaptation and a review of current adaptation approaches. Section 3 proposes an integrated solution. Section 4 provides some concluding thoughts and a look ahead.

2 Factors that Influence Adaptation

Most of the work on self-adapting software systems takes inspiration from control theory and machine learning. Control-theory splits the world into a controller and a plant. The controller is responsible for sending signals to the plant, according to a control law, so that the output of the plant follows a reference (the expected ideal output). Figure 1 shows a typical control loop. Although it is difficult to anticipate when and how change occurs in software systems, it is possible to control when and how the adaptation should react to change.

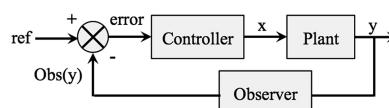


Fig. 1. Dynamic physical system

Dynamic adaptive systems require information about the running application as well as control protocols to be able to reconfigure a system. For example, keeping web services up and running for a long time requires collecting of information about the current state of the system, analyzing that information to diagnose performance problems or to detect failures, deciding how to resolve the problem (e.g., via dynamic load-balancing or healing), and acting on those decisions. Figure 2(a) shows the control process for a software system equivalent of the physical system shown in Fig. 1. The *controller* maps onto an *adaptation* process that reconfigures the runtime system to address the changing needs in its application context. Figure 2(b) show how the adaptation process can be improved using validation. Validation tracks, assesses and adjusts adaptations to ensure that they reflect user expectations.

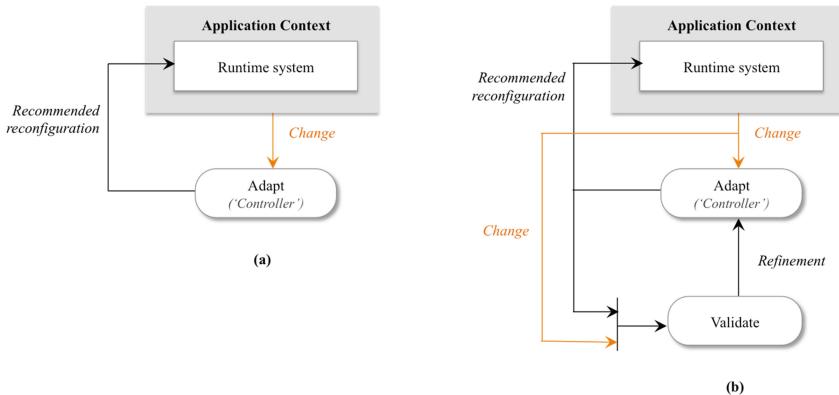


Fig. 2. Dynamic software system

Lemos et al. [7] highlight the importance of understanding the factors that influence adaptation. They posit that this helps in the comprehension of how software processes change when developing self-adaptive systems. The key challenges with current approaches include defining models that can represent a wide range of system properties, the need for feedback control loops in the life cycle of self-adaptive systems and self-validation. The nature and quality of runtime adaptation in service-oriented systems is influenced by system changes (i.e. adaptation triggers), the nature of the application and the logical area where it executes (i.e. application context), the strategy used to reconfigure the system in a particular change context (i.e. adaptation model), and the effectiveness of the adaptation (i.e. validation). Together, these factors, represent the *what*, *where*, *when* and *how*, and *right* of runtime adaptation. It is also important to note that these factors constantly shift and evolve making it difficult to specify adequate adaptation rules in advance.

2.1 Change Trigger (*What*)

A change trigger represents *what* causes adaptation and the reason for it. Change triggers are a function of changes in the business environment, service failure, and changes in the system quality and its runtime environment.

- *Business Environment Triggers.* Changes in the business environment that the system supports may trigger adaptation. This may be caused by changes in user requirements, business rules or platform. Zeng [8] describes an adaptation approach that accepts changes in user requirements and business rules on the fly and composes services to address them. Similarly Cubo [6] describes an approach that uses changes in the system context and platform to trigger adaptation.
- *Service Quality Triggers.* Failures in provided services, for example, incompatibilities, network outages and poor quality, can trigger adaptation. The quality of a service-oriented system depends not only on the quality of service (QoS) provided by services, but on the interdependencies between services and resource constraints imposed by the runtime environment. This type of corrective adaptation is typical of self-healing systems. Robinson et al. [10] describe an approach that uses a consumer-centred, pluggable brokerage model to track and renegotiate service faults and changes. The framework provides a service monitoring system, which actively monitors the quality of negotiated services for emergent changes, SLA violations and failure. A similar approach, *The Personal Mobility Manager*, described in [29] emphasizes the need for automatic system diagnosis to detect runtime errors. It helps car drivers find the best route in or between towns, by suggesting optimal combinations of transportation according to local situations, such as traffic level, weather conditions and opening hours.
- *Runtime Environment Triggers.* Changes in the runtime system can also trigger adaptation. Interacting services may impose dependability as well as structural constraints on each other (e.g. performance, availability, cost, and interface requirements). Dustdar et al. [9] describe a self-adaptation technique for managing the runtime integration of diverse requirements arising from interacting services, such as time, performance and cost. Swaminathan et al. [5] propose an adaptation approach based on self-healing as a means for addressing runtime system errors. Runtime resource contentions between services in the orchestration platform can result in significant falls in service quality. This emergent quality of service is difficult to anticipate before system composition, as resource demands are often dynamic and influenced by many factors. Newman and Kotonya [11] proposes a resource-aware framework that combines resource monitoring with dynamic service orchestration to provide a runtime architecture that mediates resource contentions in embedded service-oriented systems.

Effective adaptation must address the real cause rather than the symptom. Taiani [13] describes this as a key challenge in adaptive fault tolerant computing. Moyano et al. [14] describe a system that monitors service failure and runtime environment triggers. These are changes in hardware and firmware, including the unpredictable arrival or disappearance of devices and software component. For example, a low memory trigger may be the result of an SLA violation or runtime environment resource failure.

The resolution to the problem might involve replacing the service with a more efficient alternative or optimizing the runtime environment, or both. It is important that the adaptation process is not only able to find a good fit for the problem, but the right fit.

2.2 Application Context (*Where*)

An application context defines nature of the application and the logical area where it executes. It helps us understand *where* adaptation takes place and the constraints involved. Cubo et al. [6] discuss the importance of creating adaptive systems sensitive to their application context (i.e. domain, location, time and activity). Tanaka and Ishida [32] identify an input language and a target language as the application context for a language translation application. Most of the approaches surveyed in this paper were concerned with specific application contexts. Zeng et al. [8], for example, describe a runtime approach for supporting business change in the automotive industry. Newman [11] describes an adaptation framework for embedded resource-constrained environments. Baresi et al. [15] describes an adaptation framework for a smart home system. In their description for the *DigiHome* architecture Romero et al. [41] discuss the integration of multi-scale entities. In the *DigiHome* scenario, they consider several heterogeneous devices that generate isolated events, which can be used to obtain valuable information and to make decisions accordingly. They make use of Complex Event Processing (CEP), to find relationships between a series of simple and independent events from different sources, using previously defined rules. In their work Romero et al., use several heterogeneous devices that generate isolated events to obtain valuable information and to make decisions accordingly.

A few other approaches, including Swaminathan et al. [5], Cardellini et al. [16], and Zeng et al. [17] propose generic application contexts, but they only provide sketchy implementation details. Some approaches promote context variability. For example, Swaminathan et al. describe a context-independent, self-configuring, self-healing model for web services. However, the author provides no information about the implementation or evaluation of the model. Huang et al. [20] describe an approach for developing self-configuring services using service-specific knowledge. They evaluate their approach on three different systems (i.e. a video streaming service, an interactive search service, and a video-conference service). However, it is evident from their discussion that the context needs to be known before the application is deployed.

2.3 Adaptation Model (*When and How*)

An adaptation model shows *when* the adaptation process is carried out and *how* the model is implemented in relation to the system it manages. A decision on when to conduct adaptation is arrived at depending on when the adaptation requirements are known as well as the availability of the requirements for adaptation. If the requirements are known before the system is running then adaptation can take place at design-time, for instance introducing support for a new network communication protocol, or adding new attributes to a data model. However, if the requirements are only known after the system has started executing then adaptation will take place at runtime. This is the typical situation in ubiquitous and mobile computing scenarios. The availability of the

requirements for adaptation, such as system resources can also determine when to conduct adaptation. For example, if the resources are available online then dynamic adaptation can be conducted; otherwise it can be pushed to a later time when they will be available.

Support for adaptation is required statically, where the applications could be taken offline and adapted, and dynamically where going offline is not an available option. An adaptation model may be implemented to carry out adaptation at design-time or at runtime. Most of the work reviewed in Table 1 focuses on runtime approaches to adaptation. There is no apparent attempt to integrate design-time adaptation approaches with runtime adaptation despite some of the benefits presented by design-time adaptation. Papazoglou et al. [18] and Baresi et al. [3] identify the key techniques that can be used to achieve runtime adaptation as self-configuring, self-healing, and self-optimizing techniques.

- *Self-Configuring* is the automatic re-composition of services to adapt to changes in the service environment. The work of [19, 8, 21] describe self-configuring adaptation techniques.
- *Self-Optimizing* is the automatic re-composition of services to improve quality of a service. The work of [9, 17, 13] describes self-optimizing adaptation techniques.
- *Self-Healing* is the automatic re-composition of services to address a service failure. Self-healing techniques detect system malfunctions and initiate policy based corrective actions without disrupting the runtime environment [18].

Romay's [21] review of self-adaptation techniques in SOA reveals that current research focuses largely on self-configuring techniques. There is very little research on self-optimizing or self-healing techniques. Bucciarone et al. [22] note that focusing on only one technique limits the effectiveness of the approach.

2.3.1 Predictive vs. Reactive Models

Adaptation can also occur in response to anticipated changes (predictive) or in response to change trigger (reactive). Tanaka and Ishida [32] propose a model that focuses on predicting the executability of services (i.e. if a message request will cause execution failure). Unfortunately they provide limited detail on the implementation and evaluation of their approach. Wang et al. [24] proposes a predictability model based on the Q-Learning algorithm using the Markov Decision Processes (MDP). They explain that human oriented services are rarely predictable. They point out that many service properties keep changing in a manner that prior knowledge of these changes may not be available. Instead they suggest incorporating reinforced learning in adaptation techniques to ensure that adaptation techniques remain relevant. Their model uses a decision process that maximizes the expected sum of rewards. While predictive adaptation shows some promise, there is very little research on them.

2.3.2 Model Implementation – Embedded vs. Pluggable

In embedded adaptation, the process of monitoring and re-composition are an integral part the adaptive system. In pluggable approaches, data is collected from the running target system with non-invasive probes that report the raw data to an adaptation module that is plugged onto the system. Most of the adaptation approaches in Table 1 are

Table 1. Summary of adaptation approaches

Approach	Adaptation trigger			Adaptation model				Validation	Application Context (G = Generic S = Specific)
	Runtime environment	Service quality	Business environment	Predictive (P) Reactive (R)	Embedded (E) Pluggable (P)	Design-time support (D) Runtime support (R)			
Zeng et al. [8]	○	○	●	R	E	R	○	S	
Swaminathan [5]	○	●	●	R	N/A	R	○	G	
Cubo et al. [6]	○	○	●	R	E	R	○	S	
Huang et al. [20]	○	○	●	R	E	R	○	S	
Dustdar et al. [9]	●	○	●	R	E	R	○	S	
Autili et al. [28]	●	○	●	R	E	R	●	S	
Cardellini et al. [16]	●	●	●	R	P	R	●	G	
Lorenzoli et al. [29]	○	●	○	R	P	R	○	S	
He et al. [30]	○	●	○	R	E	R	●	S	
Mateescu et al. [31]	○	○	●	R	P	R	○	S	
Zeng et al. [17]	●	●	●	P	E	R	○	G	
Robinson et al. [10]	○	●	○	R	P	R	○	S	
Tanaka et al. [32]	○	○	●	P	E	R	○	S	
Siljee et al. [33]	○	●	●	R	E	R	○	S	
Orriens et al. [34]	○	○	●	R	E	R	○	S	
Wang et al. [24]	○	○	●	R	E	R	○	S	
Sliwa et al. [35]	○	○	●	R	E	R	○	S	
Lin et al. [36]	○	○	●	R	E	R	○	S	
Ivanovic et al. [12]	●	●	●	P	E	R	○	S	
Hussein et al. [37]	○	●	○	R	P	R	○	S	
Hirschfield et al. [38]	○	●	○	R	E	R	○	S	
Totic et al. [39]	○	●	○	R	P	R	○	S	
Maurer et al. [40]	○	●	○	R	P	R	○	S	
Romero et al. [41]	○	○	●	R	P	R	○	S	
Li et al. [42]	●	●	○	R	P	R	○	S	
Newman et al. [11]	●	○	○	R	P	R	○	S	
Motahari-Nezhad et al. [43]	○	○	●	R	E	R	○	S	
Cugola et al. [44]	○	●	○	R	E	R	○	S	
Andre et al. [46]	●	●	●	R	P	R	○	G	

Key

● - Supported

● - Weakly Supported

○ - Not Supported N/A - Not Applicable

embedded, limiting their portability. The work of Zeng et al. [8] and Cubo et al. [6] are typical examples of this. Garlan et al. [23] state that the use of external control mechanisms for self-adaptivity is a more effective engineering solution than localizing the solution. A pluggable engine can be reused across different systems.

2.4 Adaptation Effectiveness (*Right*)

A typical adaptation process uses a predefined decision model to select an appropriate adaptation in response to a change trigger. This relationship is often stored as a set of fixed adaptation rules. However, the dynamic nature of service-oriented systems means these factors are constantly changing, which makes it difficult to specify adequate adaptation rules *a priori*. This is further complicated by the likelihood of competing adaptation requests. This means that rules used to inform adaptation decisions cannot be static and must constantly evolve to remain relevant. Most approaches that support runtime adaptation are based on rules that reconfigure systems based on fixed decision points. This means that most adaptations in service-oriented systems are responses to change rather than anticipation. One way to address the problem is through the validation of adaptation decisions. Validation serves two key roles. First, it provides a mechanism for assessing the effectiveness of an adaptation decision i.e. how well a recommended adaptation addresses the concerns for which the system is reconfigured. Secondly it provides us with insights into the nature of problems for which different adaptations are suited.

Most autonomic systems are underpinned by IBM's *Monitoring, Analysis, Planning, and Execution* model (MAPE) [47]. However, while the model is evident in many self-adaptive frameworks for SOA, it lacks a runtime mechanism for supporting validation. The analysis phase of the MAPE model performs reasoning based on the details provided by the monitoring phase. The goal is to arrive at a decision on whether adaptation is required. The planning phase then identifies the appropriate adaptation and the execution phase changes the behavior of the managed resource. The entire process is based on predefined rules.

None of the approaches surveyed in this paper provided integral support for validation. However, several researchers underscored its centrality to effective adaptation [18, 45]. Skalkowski et al. [25] describe how a clustering algorithm can be used to provide automatic recognition of similar system states and grouping them into subsets (called clusters). Schumann and Gupta [26] propose a validation method to calculate safety regions for adaptive systems around the current state of operation based on Bayesian probabilities. Canfora et al. [27] propose regression testing to check the evolution of service-oriented systems.

2.5 Summary

Table 1 shows our results of surveying 29 approaches that provide runtime adaptation for SOS's. It is important to mention that the survey was intended to be representative rather than exhaustive. The approaches were carefully selected to provide a good coverage of current adaptation approaches. Each approach is reviewed in terms of the nature and extent of support for change triggers, adaptation model, validation and application context. Most of the approaches provide limited support for runtime and service quality triggers. However, they provide comparatively good support for business environment triggers. Only Ivanovic et al. [12] describes an approach for supporting all three adaptation triggers. In their work they talk of the computational cost of service networks as being dependent on internal and external factors. They recognize

that triggers for adaptation are due to overlapping factors that are both internal and external to the service. Of the approaches reviewed, only provide support for adaptation validation. However, the support is very limited. There is poor support for diversity with most approaches designed to support specific application contexts. This limitation may be related to the fact that most of the approaches are embedded.

3 Support for Validation in Runtime Adaptation

We have developed a consumer-centred, pluggable runtime adaptation framework that integrates and extends the strengths of current approaches to support validation [48]. Figure 3 shows the architecture of the framework. The *sensor sub-system* is responsible for monitoring the system for changes and for the initial decision phase of the adaptation process. When a system change or changes match the conditions specified on a sensor, the *sensor manager* invokes the adaptation process by passing it the change information. The *adaptation manager* is responsible for assessing the request and recommending a suitable adaptation solution.

The *validation sub-system* uses machine-learning algorithms to assess past adaptation decisions and to modify the rules that inform the decisions. Figure 3 shows the validation process. We use clustering algorithms to assess and refine the adaptation decisions and deep learning to review and improve the long-term accuracy of predictions. Our framework uses the past behavior of consumers rather than their opinion

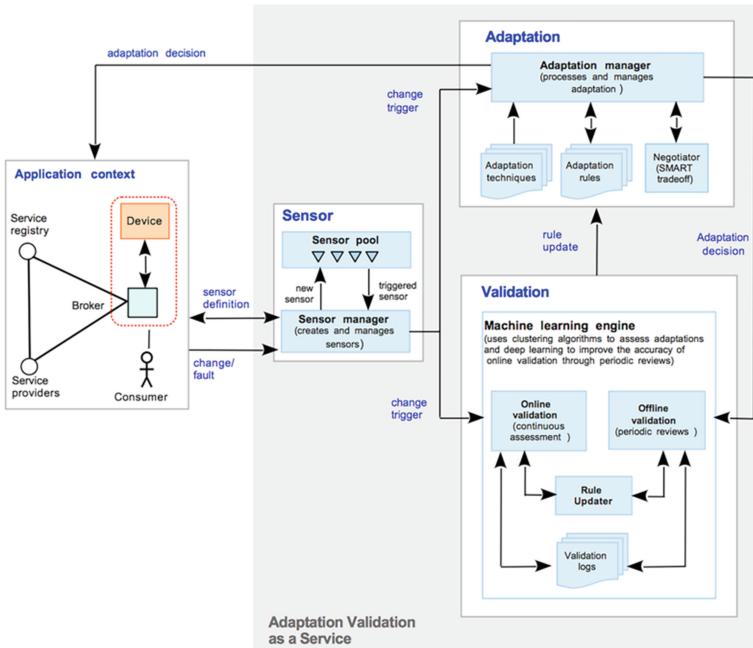


Fig. 3. Adaptation framework with support for validation

to gauge reputation and quality of feedback. If similar users in the recent past have repeatedly accepted a particular adaptation decision then the decision is most likely valid. The user is not prompted for feedback and therefore they cannot intentionally manipulate the feedback they provide. Decision logs record the change triggers received and the adaptation decision made. Machine learning algorithms then assess the user adaptation decisions against adaptation triggers and form, and refine rules that inform future adaptation decisions.

4 Conclusions

The paper has discussed the importance of runtime adaptation in SOS's and identified the factors that influence it. These factors describe the *what, where, when and how* and *right* of adaptation. Specifically adaptation triggers tell us what cause adaptation, the application context tells us where to adapt, the adaptation models tell us when and how to adapt and validation tells us how effective the adaptation is.

We have used these factors to review the current state of runtime adaptation in service-oriented systems. Our survey reveals that most of the approaches provide patchy support for the key factors that influence adaptation. Most adaptation approaches are tied to particular application contexts, focus on specific aspects changes and are embedded in the application they manage. It is also clear that there is limited empirical evidence to indicate the effectiveness of the approaches reviewed. Lastly, we have provided a possible solution that integrates and extends the strengths of current approach to support validation. We believe this paper makes a significant contribution towards understanding and addressing a challenging problem.

References

1. Erl, T.: SOA Principles of Service Design. Prentice Hall, Upper Saddle River (2008)
2. Rosen, M., Lublinsky, B.K., Smith, T., Balcer, M.J.: Applied SOA: Service-Oriented Architecture and Design Strategies. Wiley, Hoboken (2008)
3. Baresi, L.: Self-adaptive systems, services, and product lines. In: Proceedings of the 18th International Software Product Line Conference SPLC, Florence, Italy, September 15–19, pp. 2–4 (2014)
4. Taylor, R.N., Medvidovic, N., Oreizy, P.: Architectural styles for runtime software adaptation. In: European Conference on Software Architecture, Joint Working IEEE/IFIP Conference European Conference on Software Architecture, Cambridge, pp. 171–180 (2009)
5. Swaminathan, R.K.: Self configuring and self healing web services. In: Complex Software Systems, CECS IT project report, Part of the Special Projects Group. University of Waterloo, Waterloo (2008)
6. Cubo, J., Canal, C., Pimentel, E.: Supporting context awareness in adaptive service composition. In: Proceedings of 1st Workshop on Autonomic and SELF-Adaptive Systems (WASELF 2008), (JISBD 2008), pp. 64–73, SISTEDES, Gijón, Spain (2008)

7. de Lemos, R., et al.: Software engineering for self-adaptive systems: a second research roadmap. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software engineering for self-adaptive systems II. LNCS, vol. 7475, pp. 1–26. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_1
8. Zeng, L., Benatallah, B., Lei, H., Ngu, A., Flaxer, D., Chang, H.: Flexible composition of enterprise web services. *Electron. Mark.* **13**(2), 141–152(12) (2003)
9. Dustdar, S., Goeschka, K., Truong, H., Zdun, U.: Self-adaptation techniques for complex service-oriented systems. In: 5th International Conference Next Generation Web Services Practices, pp. 37–43 (2009)
10. Robinson, D., Kotonya, G.: A runtime quality architecture for service-oriented systems. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 468–482. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_35
11. Newman, P., Kotonya, G.: Managing resource contention in embedded service-oriented systems with dynamic orchestration. In: Proceedings of 10th International Conference on Service Oriented Computing, pp. 435–449 (2012)
12. Ivanovic, D., Carro, M., Hermenegildo, M.: Towards data-aware QoS-driven adaptation for service orchestrations. In: IEEE International Conference on Web Services, pp. 107–114 (2010)
13. Taiani, F., Fabre, J.: Some challenges in adaptive fault tolerant computing. In: 12th European Workshop on Dependable Computing, Toulouse, France, p. 3 (2009)
14. Moyano, F., Baudry, B., Lopez, J.: Towards trust-aware and self-adaptive systems. In: Fernández-Gago, C., Martinelli, F., Pearson, S., Agudo, I. (eds.) Trust Management VII. IFIPTM 2013, IFIP Advances in Information and Communication Technology, vol. 401, pp. 255–262. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38323-6_20
15. Baresi, L., Guinea, S., Pasquale, L.: Service-oriented dynamic software product lines. *Computer* **45**, 42–48 (2012)
16. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: Towards self-adaptation for dependable service-oriented systems. In: de Lemos, R., Fabre, J.C., Gacek, C., Gadducci, F., ter Beek, M. (eds.) Architecting Dependable Systems VI. LNCS, vol. 5835, pp. 24–48. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10248-6_2
17. Zeng, L., Lingenfelder, C., Lei, H., and Chang, H.: Event-Driven quality of service prediction. In: Proceedings of the 6th International Conference on Service-Oriented Computing, pp. 147–161, Sydney (2008)
18. Papazoglou, M.P., Traverso, P., Distar, S., Leymann, F.: Service oriented computing: state of the art and research challenges. *IEEE Comput. Soc.* **40**, 38–45 (2007)
19. Madkour, M., El Ghanami, D., Maach, A., Hasbi, A.: Context-aware service adaptation: an approach based on fuzzy sets and service composition. *J. Inf. Sci. Eng.* **29**, 1–16 (2013)
20. Huang, A., Steenkiste, P.: Building self-configuring services using service specific knowledge. In: Proceedings of the IEEE International Symposium on High Performance Distributed Computing, pp. 45–54. IEEE Computer Society, Washington DC, USA (2004)
21. Romay, M., Fernández-Sanz, L., Rodríguez, D.: A systematic review of self-adaptation in service-oriented architectures. In: Proceedings of the Sixth International Conference on Software Engineering Advances, Barcelona, Spain, pp. 331–337 (2011)
22. Bucciarone, A., Marconi, A., Mezzina, C.A., Pistore, M., Raik, H.: On-the-fly adaptation of dynamic service-based systems: incrementality, reduction and reuse. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) Service-Oriented Computing, ICSOC 2013. LNCS, vol. 8274, pp. 146–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_11

23. Garlan, D., Schmerl, B., Chang, J.: Using gauges for architecture-based monitoring and adaptation. In: Proceedings of Working Conference on Complex and Dynamic System Architecture, Brisbane, Australia (2001)
24. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) Service-Oriented Computing, ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17358-5_7
25. Skalkowski, K., Zieliński, K.: Automatic adaptation of SOA systems supported by machine learning. In: Camarinha-Matos, L.M., Tomic, S., Graça, P. (eds.) Technological Innovation for the Internet of Things, DoCEIS 2013, IFIP Advances in Information and Communication Technology, vol. 394, pp. 61–68. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37291-9_7
26. Gupta, P., Schumann, J.: A tool for verification and validation of neural network based adaptive controllers for high assurance systems. In: Proceedings of High Assurance Software Engineering (HASE). IEEE (2004)
27. Canfora, G., Di Penta, M.: SOA: testing and self-cheking. In: Proceedings of the International Workshop on Web Services Modeling and Testing, pp. 3–12, Palermo, Italy, June 2006
28. Autili, M., et al.: A development process for self-adapting service oriented applications. In: Krämer, B.J., Lin, K.J., Narasimhan, P. (eds.) Service-Oriented Computing – ICSOC 2007. LNCS, vol. 4749, pp. 442–448. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74974-5_41
29. Lorenzoli, D., Mussino, M.P., Sichel, A., Tosi, D.: A SOA based self-adaptive personal mobility manager. In: Proceedings of IEEE International Conference on Services Computing, pp. 479–486 (2006)
30. He, Q., Yan, J., Jin, H., Yang, Y.: Adaptation of web service composition based on workflow patterns. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) Service-Oriented Computing – ICSOC 2008. LNCS, vol. 5364, pp. 22–37. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_6
31. Mateescu, R., Poizat, P., Salaün, G.: Adaptation of service protocols using process algebra and on-the-fly reduction techniques. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) Service-Oriented Computing – ICSOC 2008. LNCS, vol. 5364, pp. 84–99. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_10
32. Tanaka, M., Ishida, T.: Predicting and learning executability of composite web services. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) Service-Oriented Computing – ICSOC 2008. LNCS, vol. 5364, pp. 572–578. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_48
33. Siljee, J., Bosloper, I., Nijhuis, J., Hammer, D.: DySOA: making service systems self-adaptive. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 255–268. Springer, Heidelberg (2005). https://doi.org/10.1007/11596141_20
34. Orriens, B., Yang, J., Papazoglou, M.: A rule driven approach for developing adaptive service oriented business collaboration. In: Benatallah, B., Casati, F., Traverso, P. (eds.) Service-Oriented Computing - ICSOC 2005. LNCS, vol. 3826, pp. 182–189. Springer, Heidelberg (2005). https://doi.org/10.1007/11596141_6
35. Sliwa, J., Gleba, K., Amanowicz, M.: Adaptation framework for web services provision in tactical environment. In: Military Communications and Information Systems Conference, Wrocław, Poland, pp. 52–67 (2010)
36. Lin, K., Zhang, J., Zhai, Y., Xu, B.: The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA. In: Service Oriented Computing and Application, pp. 157–168. Springer, Heidelberg (2010)

37. Hussein, M., Gomaa, H.: An architecture-based dynamic adaptation model and framework for adaptive software systems. In: 9th IEEE/ACS International Conference, pp. 165–172 (2011)
38. Hirschfeld, R., Kawamura, K.: Dynamic service adaptation. In: Proceedings of Distributed Computing Systems Workshops, pp. 290–297 (2004)
39. Totic, V., Ma, W., Pagurek, B., Esfandiari, B.: Web Service Offerings Infrastructure (WSOI) - a management infrastructure. In: Proceedings of NOMS. IEEE/IFIP Network Operations and Management Symposium, Seoul, South Korea, pp. 817–830 (2004)
40. Maurer, M., Brandic, I., Emeakaroha, V., Dustdar, S.: Towards knowledge management in self-adaptable clouds. In: Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems (SEASS 2010), Miami, Florida, USA, pp. 527–534 (2010)
41. Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R., Eliassen, F.: The Digihome service-oriented platform. *Softw.: Pract. Exp.* **43**(10), 1205–1218 (2013)
42. Li, G., Liao, L., Song, D., Wang, J., Sun, F., Liang, G.: A self-healing framework for QoS-aware web service composition via case-based reasoning. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) *Web Technologies and Applications, APWeb 2013. LNCS*, vol. 7808, pp. 654–661. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37401-2_64
43. Motahari-Nezhad, H.R., Bartolini, C., Graupner, S., Spence, S.: Adaptive case management in the social enterprise. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) *Service-Oriented Computing, ICSOC 2012. LNCS*, vol. 7636, pp. 550–557. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34321-6_39
44. Cugola, G., Ghezzi, C., Pinto, L.S., Tamburrelli, G.: Adaptive service-oriented mobile applications: a declarative approach. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) *Service-Oriented Computing, ICSOC 2012. LNCS*, vol. 7636, pp. 607–614. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34321-6_46
45. Salehie, M., Tahvildari, L.: Self-adaptive software: landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.* **4**(2), 14 (2009)
46. Andre, F., Daubert, E., Gauvrit, G.: Towards a generic context-aware framework for self-adaptation of service-oriented architectures. In: International Conference on Internet and Web Applications and Services (ICIW 2010), pp. 309–314 (2010)
47. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, pp. 13–23. IEEE Press (2015)
48. Mutanu, L., Kotonya, G.: Consumer-centred validation for runtime adaptation in service-oriented systems. In: IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA 2016), pp. 16–23 (2016)



An End-to-End Security Model for Adaptive Service-Oriented Applications

Takoua Abdellatif^{1(✉)} and Marius Bozga²

¹ Tunisia Polytechnic School, SERCOM,
University of Carthage, 2078 La Marsa, Tunisia

Takoua.Abdellatif@ept.rnu.tn
² Univ. Grenoble Alpes, CNRS, 38000 Grenoble, France
Marius.Bozga@imag.fr

Abstract. In this paper, we present E2SM, an End-to-End Security Model and a set of algorithms to protect data confidentiality in complex adaptive Service-Oriented Applications (SOA). Starting from initial and intuitive business security constraints' settings, E2SM synthesizes a complete security configuration that is formally verified. E2SM is adapted to dynamic security constraints' modifications and to services' architecture reconfiguration. Thanks to its compositional verification, only impacted services' security is rechecked which makes E2SM suitable to adaptive and scalable SOA.

Keywords: Configuration synthesis · Security formal checking
SOA · Adaptation

1 Introduction

SOA applications are generally built from existing services and based on standard tools for service composition like Web Services (WS) [1]. These applications are deployed in various domains like e-health systems, e-commerce and social networks. They exchange critical information between mistrusted parties and networks and have to preserve people's privacy and business data confidentiality. Several standards and tools [2–4] are currently used to secure WS communications. They are generally based on access control policies that focus on point-to-point communications. Nevertheless, end-to-end security is required and implies tracking information flow through all the system services and analyzing data dependencies. Typically, forwarding confidential data to unauthorized services has to be detected. Consequently, data dependence has to be tracked and explicitly exposed; otherwise, security composition can induce security leaks called interference [5].

An application free of this problem is said to satisfy the **non-interference** property [8]. To check that a WS satisfies non-interference, a classical technique consists in classifying WS data with respect to its security level and verifying that data with low-level security constraint (like public data) is not influenced or calculated from any data with high-level security constraint (like confidential data). Furthermore, the

security classification model has to consider authorization rules and trust between services and consider the network not unique as potential attacker. Indeed, current SOA applications are generally composed of loosely coupled services that come from different providers. Tracking data flow to non-authorized services is not an easy task in a distributed system [6].

Standard orchestration languages like BPEL (Business Process Execution Language) [7] are practical to build complex composed WS and to track their execution for reliability reasons mainly. Nevertheless, these languages, as well as WS security standards, provide no way to define data security classification and to check non-interference property. There is a clear need for a new security model and a **practical** tool that helps administrators setting intuitive security configuration parameters and synthesizes automatically the whole security configuration. The tool has to be **robust** proving formally that the security configuration satisfies security policies. It has to be **scalable**, that is, when the number of services' activities is huge or when the number of composed services is important, security configuration is calculated within an acceptable reasonable time. Furthermore, security configuration has to be **adaptable** to special cases like emergency where security constraints have to be relaxed. For example, in a home gateway application, if a fire is detected, the information has to reach a remote monitor service in a short time even though such information is not protected with cryptography encryption. More generally, a compromise between security constraints and functionality needs has to be found and relaxing some information secrecy is sometimes needed to reach this compromise. This feature is called declassification and is still a hard problem in security research field and is rarely implemented in real applications like SOA [9, 10]. Another aspect of adaptation is when the system architecture is modified. Typically, removing or adding new services or modifying some security configuration settings may require the security configuration synthesis that has to be executed rapidly not to slow down the adaptation time.

In this paper, we propose E2SM (End-to-End Security Model), a security model for composed and adaptive SOA applications constructed as a composition of BPEL processes. The main idea of E2SM is to abstract BPEL processes to a set of graphs to show data dependencies and starting from a few initial configuration settings, the whole system configuration is generated while checking its non-interference property. Non-interference checking is modular: to check that a hierarchical BPEL process is secure, it is sufficient to check that each involved BPEL process is secure. Furthermore, E2SM assists users (service designers, developers or administrators) to set-up business oriented security constraints at a high level, checks constraints coherence and synthesizes a complete secure configuration. If the user's security constraints induce non-secure configuration, the user is guided to correct his initial configuration by relaxing some constraints or modifying his initial configuration. Dynamically, E2SM checks any security re-configuration. It tracks the affected service parts and regenerates the new security configuration in acceptable time. Our solution is practical since users do not need to be security experts, robust since the security model and algorithms are based on a formal modeling, scalable thanks to the compositional checking and adaptable to SOA dynamic reconfiguration.

The rest of the paper is structured as follows. Section 2 describes program dependence graphs as abstractions for BPEL workflows. Section 3 presents the security

model and E2SM configuration synthesis algorithms. Section 4 evaluates E2SM mainly the algorithm performance. Section 5 presents related work and Sect. 6 concludes the paper and presents its perspectives.

2 Dependence Graph Abstraction

In this section, we describe program dependence graphs (PDG) as abstraction for BPEL processes for data tracking and the system dependence graphs (SDG), our extension to PDG to abstract composed BPEL processes.

2.1 Program Dependence Graphs

Program dependence graphs (PDG) are a standard tool to model information flow through a program [11]. Graph nodes represent program statements or expressions. A data dependence edge, represented with an arrow, $x \rightarrow y$ means that statement x assigns a variable which is used in statement y (without being reassigned underway). A control dependence edge $x \rightarrow y$ means that the mere execution of y depends on the value of the expression x (which is typically a condition in an if- or while-statement). A path $x \rightarrow *y$ means that information can flow from node x to node y . Contrarily, if no path exists from node x to y , it is guaranteed there is no information flow from x to y . To identify all statements influencing a node y , the backward slice is defined as $BS(y) = \{x | x \rightarrow *y\}$. PDG is classically used in imperative languages like Java [8, 11]. We implemented in a previous work [18] a PDG generator for BPEL processes. A BPEL process is composed mainly of two types of activities, that are (1) basic activities, such as receive, reply, invoke, assign, throw, exit, and (2) structured activities, such as sequence, if, while, repeat – until, pick, f low. First, a BPEL control dependence sub-graph is constructed where nodes represent BPEL activities and edges represent possible execution sequences of the activities. This is applicable mainly for condition activities like $< \text{if....} >$ or $< \text{switch...} >$. For example, a control flow edge $x \rightarrow y$ means that the activity represented by y may execute immediately after the execution of the activity represented by x . Second, a data dependence analysis is performed attributing the system variables to their activities and based on a Definition-Use relation. For each ordered pair (n_d, n_u) , where a statement called n_d contains a definition of a variable v and used in a statement n_u , a data dependence is identified [12]. For example, Fig. 1 illustrates the graphical representation of the BPEL process of a laboratory service deployed on a cloud. It receives a patient medical record and following the record type, it forwards the record to the radio laboratory service or the blood laboratory service. The activities are indexed with their node number.

For example, node (01) corresponds to the receive BPEL activity and node (02) corresponds to the copy instruction of the assign activity. In (01), variable *input_from_reception* is used and in (02) this variable is copied to the newly defined record variable. Since there is a Definition-Use relation between the two activities w.r.t input from reception variable, a graph edge is created between node (01) and (02). Similarly, there is an edge between nodes (01) and (03). Node (04) corresponds to a structured *if activity* where variable record type is used. The Definition-Use relation

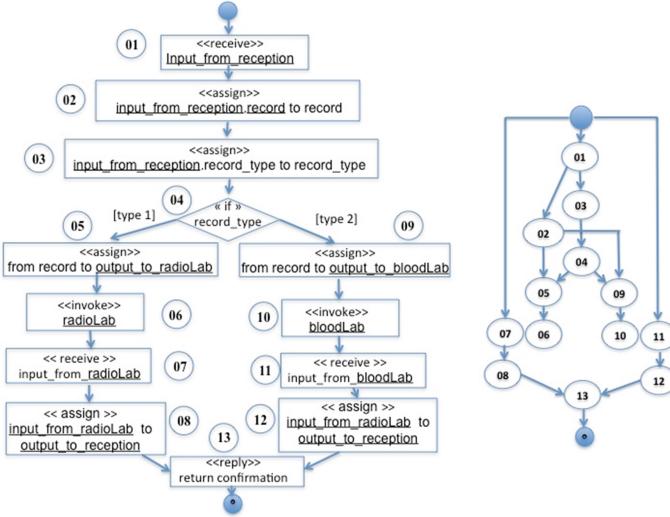


Fig. 1. Graphical representation of the Laboratories BPEL and the correspondent PDG

with respect to variable *record_type* allows creating the edge between (02) and (04). On the other hand, a control dependence creates the edge between (04) and (05) and between (04) and (05) due to the condition activity at node (04).

2.2 System Dependence Graphs

For composed BPEL processes, SDG describes the information flow for the entire system. To build it, we consider the PDG of all services and connect them with binding edges. Each binding edge corresponds to an inter-service connection: the source is the sender output endpoint and the sink is the destination service endpoint. Furthermore, for each primitive service, we construct a restricted PDG with nodes corresponding to input and output variables and edges joining the input nodes to the output nodes they depend on. Variable dependence in atomic services can be identified either using classical program dependence graphs [13] or using a description of the service behavior. In the worst case, we consider that all service outputs depend on all service inputs. Hereafter, we give a formal definition of SDG.

Definition 1 (System Dependence Graph). Let WSS be a WS system composed of n services $(S_i)_{i=1,n}$. For each service S_i , let $G_i = (N_i, E_i)$ be its associated PDG. The system dependence graph $G = (N, E)$ is constructed as follows: $N = \cup_{i=1,n} N_i$ and $E = E_{\text{inter}} \cup (\cup_{i=1,n} E_i)$ where E_{inter} is the set of edges corresponding to inter-service bindings.

3 E2SEM Security Model

Many security annotation models are proposed in the literature [5]. Nevertheless, DLM, the decentralized label model [14] initially proposed for Java programs, is more appropriate for distributed services with mutual distrust. We extend DLM for SOA and we define non-interference for BPEL-based workflows. Afterwards, we propose two algorithms for non-interference checking and configuration synthesis.

3.1 Decentralized Label Model

The essentials of DLM are objects containing information to protect, principals and labels. Objects can be variables, storage locations or input/output communication channels receiving/sending data. Principals are persons or programs that own or access pieces of system information. To express confidentiality policy, labels are associated to objects and a principal can be owner initiating the information or reader authorized to access that information. A label general structure is $L = \{o_1 : r_{11}, r_{12}, \dots; o_2 : r_{21}, r_{22}, \dots; \dots; o_n : r_{n1}, r_{n2}, \dots\}$ where o_i are owners and r_{ij} are readers. For example, in a clinical SOA, the medical record can be labeled $L = \{\text{patient} : \text{clinic, doctor}\}$ which indicates that the patient principal allows the clinic and the doctor to access his information. Labels are ordered using the no more restrictive than relation, represented by \subseteq symbol. Given two labels L_1 and L_2 , we have $L_1 \subseteq L_2$ if and only if owners of L_1 are included in L_1 and, for a given owner, the readers of L_2 are included in those of L_1 . For example, if $L_1 = \{\text{clinic} : \text{doctor1}\}$ and $L_2 = \{\text{patient} : \text{doctor1, doctor2}\}$, we have $L_2 \subseteq L_1$. During program computation, when information is labeled with L_1 and L_2 , respectively, the result should have the least restrictive label that maintains all the flow restrictions specified by L_1 and L_2 . This least restrictive label, the join of L_1 and L_2 (written as $L_1 \cup L_2$), is constructed so that owners is the union of L_1 and L_2 owners and the reader set for each owner in L_1 and L_2 is the intersection of their corresponding reader sets. For example, let us suppose that a medical record is composed of two parts so that the first part is labeled $L_1 = \{\text{patient} : \text{doctor1}\}$ and the second one is labeled $L_2 = \{\text{patient} : \text{doctor1, doctor2}\}$. The medical record has to respect both constraints and then only doctor1 is allowed to read the medical record content with label $L_1 \cup L_2 = \{\text{patient} : \text{doctor1}\}$. Note that security labels represent a lattice (L, \subseteq) [14] where L denotes the finite set of labels that is partially ordered by \subseteq . We denote by $*$ the weakest principal used to annotate public information. For example, label $\{p : *\}$ indicates that information is owned by principal p which allows all principals access its information.

Principals can delegate their information ownership to other principals with regard to a *trust* relation. According to trust hierarchy, information can be relabeled in a safe way following two forms. In the first form, label ownership is changed. A label's owner O can be replaced by owner O' so that O *trusts* O' . The second form of relabeling is declassification, which allows relaxing security constraints in a safe way. The information is copied in a fresh variable and relabeled to a less restrictive security label. This is allowed only by data owner or by another principal trusted by the owner. The new label is usually calculated from the information label by adding readers (more principals are allowed to access information) or removing some policies (owners with their respective readers), which imposes less reading constraints on that information.

For example, in the Fig. 1. use-case, the patient record labeled $L = \{\text{patient: laboratory}\}$ does not allow initially neither the blood-Lab or the radio-Lab to read data. Nevertheless, since these labs are trusted by the patient, they can be added as readers and L is declassified to $L_2 = \{\text{patient:laboratory, blood-Lab, radio-Lab}\}$.

Since we have two kinds of services in SOA, managed and external services, we extend DLM with **required** and **provided** labels. Required labels are immutable labels that represent external constraints, typically third-party service requirements that cannot be changed by the administrator. Provided labels are labels set on managed resources and can be modified by the administrator when needed.

3.2 PDG-Based Non-interference Checking

Let N be the set of PDG nodes, X the set of program variables and L the set of security labels. Let $S : N \cup X \rightarrow L$ be a function assigning security labels L to nodes in $N \cup X$ defined as follows. For a program statement represented by a node n , if a variable v is defined (that is, assigned) in that statement, then $S(n) = S(v)$. For example, in Fig. 1., we have $S((01)) = S(\text{input_from_reception})$. If no variable is assigned in the statement represented by the node n , then $S(n) = \bigcup_i L_i$ where L_i represents the security label of i^{th} variable used in that statement [11]. Non-interference property is satisfied in annotated PDG with S if and only if for every edge $x \rightarrow y$ it holds $S(x) \subseteq S(y)$. Classically, non-interference is iteratively checked starting from specific nodes called slicing criteria that we define hereafter. For a statement of interest represented by a node x , the backward slice $BS(x)$ extracts those statements that potentially have an influence onto that statement. This later is called the slicing criterion. In our work, we define slicing criteria are the nodes that have required labels. Typically, a slicing criterion is a node representing an endpoint to an external service and imposing its security constraint. Consequently, required security label specifies a limit so that only information having a smaller security label may reach that statement. Inversely, provided labels can be assigned to any node in PDG. Formally, provided labels are defined by a partial function $P : N \cup X \rightarrow L \cup \{\perp\}$. Similarly, required security is defined as a partial function $R : N \cup X \rightarrow L \cup \{\perp\}$. In this model, the security label $S(n)$ of every node n must moreover satisfy: $P(n) \subseteq S(n) \subseteq R(n)$ whenever $P(n)$ and/or $R(n)$ are defined. In this extended model, the non-interference property in an annotated PDG can be verified as follows.

Proposition 1 (Non-interference checking - version 1). In a PDG with P and R the functions assigning respectively the provided and required security labels, the non-interference property holds if the following condition is satisfied: $\forall n \in \text{dom}(R)$, $\forall x \in \text{dom}(P) \cap BS(n)$, $P(x) \subseteq R(n)$. Considering actual security labels S , the next proposition provides an alternative way for checking non-interference [11].

Proposition 2 (Non-interference checking - version 2). In a PDG with P and R the functions assigning respectively provided and required security labels, non-interference property is satisfied if the following condition holds: $\forall n \in \text{dom}(R)$, $S(n) \subseteq R(n)$.

Algorithm 1: Secure configuration without declassification

Input: $G = (N, \rightarrow)$ a PDG graph, X a set of variables, P, R provided and required security annotations on $N \cup X$

Output: S complete label assignment on $N \cup X$

```

/* Initialization */
foreach n ∈ N ∪ X do
    S(n) := {⊥}
    if P(n) defined then S(n) := P(n) endif
end
/* Main Loop */
while (S changes) do
    foreach n ∈ N do
        S(n) := S(n) ∪ ∪n' ∈ BS[n] S(n')
        if (node n defines variable x) then
            S(n) := S(n) ∪ S(x) ; S(x) := S(n) endif
    end
    foreach n ∈ N ∪ X do
        if R(n) defined and S(n) ⊂ R(n) then
            stop /* required security label exceeded at n */ endif
    end

```

Algorithm 2: Secure configuration with declassification

Input: $G = (N, \rightarrow)$ a PDG, e the root node, X a set of variables, P, R provided and required security annotations on $N \cup X$

Output: D , the set of declassified nodes

/* Declassification nodes is initially empty */

$D := \emptyset$

/* Main loop */
while (D changes) do
 /* Compute candidates for declassification */
 $Z := \{ z \in N / z \text{ defines a variable, } \exists x \in \text{dom}(P),$
 $x \in \text{BS}(\{z\}, D \cup \{e\}), \exists y \in \text{dom}(R),$
 $z \in \text{BS}(\{y\}, D \cup \{e\}), P(x) \subset R(y),$
 $\text{checkTrust}(\text{this}, P(x), R(y)) \}$

/* Interact with user to select a subset Z' for declassification */
 $Z' := \text{chooseForDeclassification}(Z)$
/* Augment the set of declassified nodes */
 $D := D \cup Z'$
end
/* Final check */
if conditions of Prop 3 hold then
 exit /* secure configuration obtained */
end

Starting from an initial annotation for provided and required security levels R and P , Algorithm 1 calculates the actual security configuration S using an iterative method. Algorithm 1 can be considered as a flavor of the classical Bellman-Ford algorithm. The number of iterations depends on relabeling occurrence which depends itself on the number of vertices, the graph connectivity and the initial label distribution.

Regarding declassification, specific nodes are selected to be declassification nodes [15]. A declassification node d has a security label $S(d)$ and a required security label $R(d)$ so that the relation $S(d) \subseteq R(d)$ is not satisfied. Declassification implies that the user authorizes lowering $S(d)$ to $R(d)$. Non-interference with declassification holds if for each path from node x to y where the relation $S(x) \subseteq S(y)$ is not true, there is a declassification node d on the path with $S(x) \subseteq R(d)$ and $S(d) \subseteq S(y)$ (assuming that there is no other declassification node on that path) [11]. Therefore, information flow control with declassification is no longer transitive. For confidentiality checking with declassification, a simple solution is to represent declassification nodes as barriers where slicing stops [16]. Barrier slices are defined as follows.

Definition 2 (Barrier slice). Let $G = (N, E)$ be a PDG, C a slicing criterion and B the set of barrier nodes. The barrier slice $BS(C, B)$ for the slicing criterion C is the set of nodes on which a node $n \in C$ (transitively) depends via a path that does not contain any node of B [16].

Checking confidentiality with declassification implies checking non-interference considering barrier slices instead of backward slices where the barrier nodes are composed of the declassified nodes and the entry node (corresponding to the entry point in the program). The slicing criterion is composed of the join of nodes with required labels and declassified nodes. Consequently, Proposition 1 can be adapted considering slices with barriers as follows.

Proposition 3 (Non-interference with declassification checking). In a PDG with P and R the partial functions assigning respectively the provided and required security labels, D a set of declassified nodes, e the root of the PDG (corresponding to the entry point of the program) and B a barrier node set where $B = D \cup \{e\}$. Non-interference with declassification is satisfied if the following condition holds:

$$\forall n \in \text{dom}(R) \cup D, \forall x \in \text{dom}(P) \cap \text{BS}(\{n\}, B), P(x) \subseteq R(n)$$

Based on this definition, we propose Algorithm 2 that helps users checking and building secure configurations with declassification. Algorithm 2 calls the *checkTrust* (S, L_1, L_2) function that, for each service S and labels L_1 and L_2 , verifies the declassification condition based on trustfulness between principals.

Proposition 4. The algorithm *Secure configuration checking* accepts secure configurations and rejects non-secure ones.

Proof. Let S be the computed configuration. For each node x with required security, we have one of the following conditions: either $S(x) \subseteq R(x)$ or x is a declassified node. Otherwise, the algorithm stops. By definition, we have a secure PDG and then S is a secure configuration.

3.3 SDG-Based Non-interference Checking

To deal with system security, we extend the definition of security configuration defined for single WS (see Definition 1) to WS composition. A security system configuration is an assignment of security labels to variables and nodes within all the PDG of services composing the WS system.

Definition 3 (Security system configuration S). Let WSS be a WS system composed of a set of n services (S_i) $i = 1, n$. For each service S_i , let $G_i = (N_i, E_i)$ be the associated PDG and let X_i the set of its variables. Let E_{inter} be the set of edges corresponding to inter-service bindings. Let L be the set of labels, X the set of all service variables and N the set of all PDG nodes associated to WSS. We define a security configuration for a WS system as a mapping $S : X \cup N \rightarrow L$ that associates security labels to variables and nodes. Moreover, we require the following three matching conditions amongst the different categories, for all i, j : (1) $\forall v \in X_i, \forall n_v \in G_i, v$ defined at $n_v \Rightarrow S(v) = S(n_v)$, (2) $\forall v \in X_i, \forall n_v \in G_i, v$ used in $n_v \Rightarrow S(v) \subseteq S(n_v)$ and (3) $\forall n \in G_i, \forall m \in G_j, (n, m) \in E_{\text{inter}} \Rightarrow S(n) = S(m)$.

The two first conditions correspond to security configuration definition for a single service. The third condition states that the nodes corresponding to binding edges have the same security label since they hold the same information. By analogy to secure PDG, we define now a secure SDG as a composition of secure PDG. For SDG, we have two kinds of barrier slices: internal barrier slices inside PDG joining input variable nodes to output variable nodes and external barrier slices that correspond to inter-service bindings. For internal slices, since PDG are assumed to be secure, we have the guarantee that information does not flow from high level labels to low level labels except for declassification nodes. For external slices, the security enforcement ensures a

matching between labels of inter-service nodes connecting PDG. Furthermore, in SDG, there is no room to declassification since it was treated locally to each PDG and for each declassified node n_i , its actual security label was assigned to its required one, that is $S(n_i) = R(n_i)$.

Definition 4 (Secure SDG). Let $G = (N, E)$ be an SDG of a WS system and $G_i = (N_i, E_i)$ the set of PDG of services composing the system for $1 \leq i \leq n$. We say that G is secure iff $\forall 1 \leq i \leq n$, G_i is a secure PDG.

We define an end-to-end secure system as a system where information do not flow from high-level sources to lower level destinations except for special destinations where the user authorizes information declassification. Information dependence is detected thanks to PDG for BPEL activities and for atomic service programs.

Definition 5 (End-to-end secure WS System). Let WSS be a WS system comprising a set of composed WS services and let S the security configuration of the system. We say that WSS is end-to-end secure if for all y, x two variables, y depends on x implies $S(x) \subseteq S(y)$.

Proposition 5. Let WSS be a WS system and G its associated SDG. If G is secure then WSS is end-to-end secure.

Proof. We prove the proposition by induction on the number n of composed services the information flows through. Consider the basic case where $n = 1$. This means that information flows inside a single composed service. Let us consider any two variables y and x so that y depends on x . That means, there exist a path in the PDG from a node n_x to a node n_y where respectively x and y are defined. Since the PDG is secure, we have $S(n_x) \subseteq S(n_y)$. By matching variable and node labels, we have $S(x) \subseteq S(y)$. Consider now that the proposition is true for any WS system composed of $n-1$ services for some $n > 1$. We prove that it is true for n composed services. Let S_1, \dots, S_n be the system services, G_1, \dots, G_n their PDGs supposed secure and consider that information flows from variables x to y , which moreover belong to separate services S_1 and S_n . Let z be the output variable calculated from x inside S_1 . Since, G_1 is secure, we have

$$S(x) \subseteq S(z) \quad (1)$$

Suppose that z is further received by S_2 . By the induction hypothesis, the system composed of $n-1$ services S_2, \dots, S_n is end-to-end secure. Then

$$S(z) \subseteq S(y) \quad (2)$$

(1) and (2) implies $S(x) \subseteq S(y)$. As a conclusion, the proposition is true for a WS system with any number n of composed services. As illustrated in Algorithm 1, inside each process, the binding between the process and the service endpoints it communicates with, is checked before checking security inside the process. Applying the previous proposition, this ensures that the system is end-to-end secure.

3.4 Non-interference Runtime Checking

At run time, non-interference re-checking occurs when a label value changes corresponding to a security constraint modification. The non-interference checking and configuration synthesis is executed for the concerned process. If the label of a binding edge changes, the other edge's label is updated and then the affected service has its security configuration re-checked and synthesized. The same logic is applied when a structural re-configuration occurs in the SOA. Indeed, adding or removing a service to the application implies adding or removing a binding and this may induce modifications to edge labels.

4 E2SM Evaluation

Scalability is ensured thanks to the composed checking and security configuration synthesis that can be performed in a parallel way. For the algorithm's performance evaluation, we generate a single source graph so that the number of vertices ranges from 5 to 4000 vertices and generates edges in a probabilistic way. We use Erdos-Renyi model to generate edges. So edges are generated with a probability p where p is set to three values: 0.1 for weakly connected graph, 0.5 for moderated connected graph and 1 for full connected graph. Without loss of generality, we consider two security levels : secret and public. We distribute randomly public and private labels on graph edges and do not consider required labels not to stop the program iteration before visiting all edges.

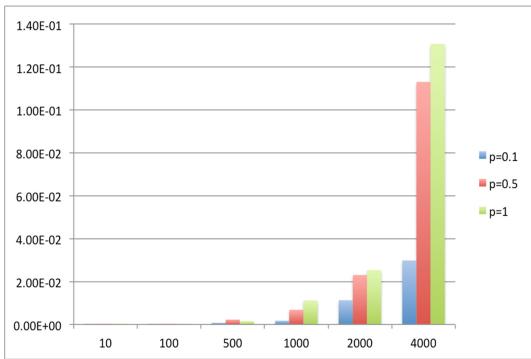


Fig. 2. Performance evaluation of configuration's synthesis program

Figure 2. shows the average time obtained after repeating the program execution 10 times. The bench is executed on a Mac OS version 10.8.5 with processor 1.3 GHz Intel Core i5 and 4 GB of Memory. The execution time increases with the number of vertices in the PDG and with the number of connectivity. We clearly see that the execution time is very acceptable and does not exceed 1 s for 4000 vertices and a fully connected PDG.

Adaptability to special cases is ensured thanks to the declassification feature, which allows a controlled relaxation of security constraints. E2SEM declassification can be extended to other kinds of declassification like temporal declassification [9]. Regarding system's reconfiguration by adding or removing a service, thanks to the compositional verification, there is no need to recheck the whole newly obtained application, only affected processes have their security configuration re-checked and synthesized.

5 Related Work

Our work is related to information flow security solutions for SOA. SEWSEC [17] is an end-to-end security tool for WS security. Compared to SEWSEC, our work allows not only non-interference checking but also security configuration synthesis, provides a formal security model and deals with adaptation. In [18], a security configuration synthesis is provided but the adopted model does not deal with declassification and adaptation. Similarly, in [19], information flow security is applied to component-based systems. Nevertheless, component code is required for label propagation, no formal model is provided and adaptation is not considered. Information flow control is also treated for event-based communications like those described in BPEL [20, 21]. For data dependence tracking, systems are modeled as Petri-nets [22, 23] or propagation graphs from workflow's log data [24]. A language based information flow is proposed in [25] to check non-interference but declassification and adaptation are not supported. In [26, 27], authors deal with chained services with no centralized orchestration service composing them. A recent work extends BPEL-orchestration engine [28] and requires BPEL processes' annotation whereas we propose a more practical approach with minimal configuration effort and configuration synthesis.

6 Conclusion

In this paper, we propose a robust security model to protect confidential data in complex business applications. Even though, we concentrate on BPEL, the work is applicable to other types of SOA composition languages. They only need to be mapped to program dependency graphs. We are currently implementing E2SM associated tool and experimenting it to secure e-health real SOA application.

References

- Walsh, A.E.: UDDI, Soap and WSDL: The Web Services Specification Reference Book. Prentice Hall Professional Technical Reference, Englewood Cliffs (2002)
- Web services security: Soap message security 1.1, February 2006. <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- Bajaj, S., et al.: Web services policy framework (wspolicy), March 2006. <http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf>
- Della-Libera, G., et al.: Web services security policy language (ws-securitypolicy), July 2005. <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
- Goguen, J.A., Meseguer, J.: Security policies and security models. In: 1982 Proceedings of the IEEE Symposium on Security and Privacy (1982)
- Zdancewic, S.: Challenges for information-flow security. In: Proceedings of the 1st International Workshop on the Programming Language Interference and Dependence, pp. 5–19 (2004)
- Alves, A., et al.: Web services business process execution language version 2.0, April 2007. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
- Ferrante, J., Ottenstein, K., Warren, J.: The program dependence graph and its use in optimization. ACM Trans. Program. Lang. Syst. **9**, 319–349 (1987)

9. Sabelfeld, A., Sands, D.: Dimensions and principles of declassification. *J. Comput. Secur.* **255–269** (2009)
10. Myers, A., Sabelfeld, A., Zdancewic, S.: Enforcing robust declassification. In: Computer Security Foundations Workshop, p. 172 (2004)
11. Snelting, G., Robschink, T., Krinke, J.: Efficient path conditions in dependence graphs for software safety analysis. *ACM Trans. Softw. Eng. Methodol.* **15**(4), 410–457 (2006)
12. Mao, C.: Slicing web service-based software. In: International Conference Service-Oriented Computing and Applications (SOCA) (2010)
13. Giffhorn, D., Hammer, C.: Precise analysis of Java programs using JOANA, pp. 267–268 (2008)
14. Myers, A.C., Liskov, B.: Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.* **9**, 410–442 (2000)
15. Hammer, C., Krinke, J., Nodes, F.: Intransitive noninterference in dependence graphs. In: Second International Symposium on Leveraging Applications of Formal Methods Verification and Validation, pp. 119–128 (2006)
16. Krinke, J.: Slicing, chopping, and path conditions with barriers. *Softw. Qual. J.* **12**, 339–360 (2004)
17. Zorgati, H., Abdellatif, T.: SEWSEC: a secure web service composer using information flow control. In: 6th International Conference on Risks and Security of Internet and Systems, Timisoara, Romania. IEEE (2011)
18. Ben Said, N., Abdellatif, T., Bensalem, S., Bozga, M.: A robust framework for securing composed web services. In: Braga, C. (eds.) FACS 2015. LNCS, vol. 9539, pp. 105–122. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28934-2_6
19. Abdellatif, T., Sfaxi, L., Robbana, R., Lakhnech, Y.: Automating information flow control in component-based distributed systems. In: ACM Sigsoft International Symposium on Component-Based System Engineering CBSE. ACM (2011)
20. Bartolini, C., Bertolino, A., Marchetti, E., Parissis, I.: Data flow-based validation of web services compositions: Perspectives and examples. *Training* 298–325 (2008)
21. Bartolini, C., Bertolino, A., Marchetti, E., Parissis, I.: Data flow-based validation of web services compositions: perspectives and examples. In: de Lemos, R., Di Giandomenico, F., Gacek, C., Muccini, H. (eds.) WADS 2007. LNCS, vol. 5135, pp. 298–325. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85571-2_13
22. Busi, N., Gorrieri, R.: A survey on non-interference with petri nets. In: Desel, J., Reisig, W. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 328–344. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_8
23. Busi, N., Gorrieri, R.: Structural non-interference, in elementary and trace nets. *Math. Struct. Comput. Sci.* **19**, 1065–1090 (2009)
24. Accorsi, R., Wonnemann, C.: Static information flow analysis of workflow models. In: BPSC (2010)
25. Hutter, D., Volkamer, M.: Information flow control to secure dynamic web service composition. In: Clark, J.A., Paige, R.F., Polack, F.A.C. (eds.) SPC 2006. LNCS, vol. 3934, pp. 196–210. Springer, Heidelberg (2006). https://doi.org/10.1007/11734666_15
26. Wei, S., I-Ling, Y., Bhavani, T., Elisa, B.: The SCIFC model for information flow control in web service composition. In: 2009 IEEE International Conference on Web Services, pp. 1–8 (2009)
27. She, W., Yen, I.L., Thuraisingham, B.: Enhancing security modeling for web services using delegation and pass-on. In: 2008 IEEE International Conference on Web Services, pp. 545–552 (2008)
28. Demongeot, T., Totet, E., Le Traon, Y.: Preventing data leakage in service orchestration. In: IAS (2011)



Runtime Migration of Applications in a Trans-Cloud Environment

Jose Carrasco, Francisco Durán^(✉), and Ernesto Pimentel

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga,
Málaga, Spain
{josec,duran,ernesto}@lcc.uma.es

Abstract. Making an application independent of the cloud provider where it is going to be deployed is still an open issue. In fact, cloud agnostic software development still presents important challenges to be solved, and one of them is the problem of runtime migration of components already deployed on a given provider to a different one. Even more difficult is dealing with the interoperability issues when the migration also implies a change of service level (*i.e.*, from IaaS to PaaS, or *vice versa*). This paper presents an algorithm for the parallel migration of cloud applications. The migration is performed component-wise, in the sense that each component of the application to be migrated may be deployed on a specific service on a specific provider, and be moved to a different provider, possibly changing the service level between IaaS and PaaS of each of them individually. Since the migration of components with state pose additional difficulties, we only consider stateless components. Our solution relies on three of the key ingredients of the *trans-cloud* approach: a unified API, agnostic topology descriptions, and mechanisms for the independent specification of providers. We show how our approach solves some of the current interoperability and portability issues of cloud environments, and allows us to provide a solution for migration. We present an implementation of our proposed solution and illustrate it with a case study and experimental results.

1 Introduction

In recent years, as an answer to market demands, Cloud Computing has experienced a growth in goals and capabilities [14]. In this quick evolution of the technology, vendors have developed their own cloud solutions by providing different approaches, where different functionalities are provided by different APIs following diverse business models. In fact, as a consequence of this heterogeneity, cloud developers will often be locked-in specific services from cloud providers, since many interoperability and portability restrictions are found when different vendors' solutions want to be used.

Recent proposals, like those in [9, 11, 13], take advantage of new advances in the management of topologies of applications that have been deployed on cloud to solve most of the interoperability issues related to application deployment.

Among these emerging solutions, we find platforms with the capability of allowing developers to find the best alternative for the deployment of each component of their applications. The last step in this direction is the possibility of deploying applications combining services from IaaS and PaaS levels, possibly by different vendors in trans-cloud environments [6]. In fact, as we can see in, e.g., [1, 5, 12], the decision of what vendor or service level to use to distribute the components during an application deployment is not trivial. It is indeed a challenge, because of the multitude of cloud offerings. Migration of individual components or entire applications may indeed be unavoidable over time, because of changes in the offered services, prices, security policies, or simply because a provider just stops providing its services. Once developers can take advantage of the features of different kinds of services, they will be interested as well in optimizing the cloud resources usage and improve their applications' performance.

If component deployment reconfiguration is a complex task, doing it at runtime, trying to minimize down times is even more challenging. Migration is still an unresolved topic which has been widespread studied by both academia and industry (see, e.g., [10, 16]). To migrate some components of a running cloud application, it is necessary to orchestrate the entire cloud context, such as services and bound resources, to reach the expected movement of the components services but taking into account the possible interoperability and portability problems. In fact, migration opens a lot of new key issues related to cloud resources, application components, and their management. We can find several proposals (see, e.g., [2, 3, 8]) on the *live migration of cloud application's components*, focusing on the movement of running application components between different vendors. However, these solutions focus with one service level (cf. [3, 8, 15]).

In [7] we proposed an orchestration algorithm to migrate a single application's stateless component between different providers in an agnostic way, what means that it is not bound to any service level, either IaaS or PaaS, of any particular provider. In this new proposal we go one step further and extend the aforementioned tool by proposing a new algorithm to reach the component-wise migration of entire applications. Since the migration of components with state pose additional difficulties, we only consider stateless components. The algorithm performs the migration in parallel and agnostically, by moving just the necessary application's components to respective target services, independently of the target providers and abstraction levels, either IaaS or PaaS.

This extension is built over *trans-cloud* concepts [6]. Specifically, the solution in [6] uses the TOSCA standard to model agnostic applications' topologies, which do not use any specifics of the target providers over which they will be deployed. The information related to the cloud service level, IaaS or PaaS, is added by means of a policy-based mechanism independent of the topology description. The trans-cloud environment processes these specifications and uses an homogenization API, which unifies IaaS and PaaS services of different vendors, to orchestrate the deployment over the required cloud services.

By relying on the trans-cloud infrastructure operations such as *stop*, *re-start*, *move* and *re-connect* are used on the necessary components independently of the

service level, IaaS or PaaS, the cloud technology or any other dependencies. The algorithm receives a set of components of an application that has to be migrated and their target locations. It operates on the application's components following concurrent policies, working in parallel.

The rest of this paper is structured as follows. Preliminaries about trans-cloud deployment and its current implementation are presented in Sect. 2. The proposed migration algorithm for stateless components is described in Sect. 3. Details on the implementation of the algorithm are presented in Sect. 4, together with some experimental results. Finally, Sect. 5 conclude the paper and presents some plans for future work.

2 An Overview of Trans-Cloud

In this section we provide an overview of trans-cloud and its main capabilities. These concepts are illustrated with a running case study, which will be later used to show the use of the our proposal in Sect. 3 and to evaluate it in Sect. 4.

2.1 A Trans-Cloud Environment

Trans-cloud significantly reduces the portability and interoperability related issues. It allows developers to deploy their application by using available vendors' services and resources, at IaaS or PaaS levels, releasing them from the usual infrastructure limitations while defining their applications. The trans-cloud approach relies on three main ideas: agnostic topology descriptions, target services specifications, and a unified API.

For the agnostic specification of applications' topologies we use the TOSCA standard,¹ which allows us to provide full-detailed specifications of application components, how they are inter-related and their respective configurations. To manage the selected cloud services to deploy applications' components, we use Apache Brooklyn, which defines a common interface for the homogeneous treatment of different cloud vendors and their services.

The description of target services needs a mechanism to agnostically specify the target cloud services which will be managed from the Unified API. To do this, we use *locations*, which represent cloud services on which to perform the deployment, both for IaaS and PaaS services. As we will see in the coming sections, locations are defined independently of the topologies, what allows us to change target services just by changing the corresponding locations.

The trans-cloud tool presented in [6] builds on the Brooklyn-TOSCA open project for enabling an independent specification of the used services, and on the Apache Brooklyn project to provide a common API for the unified management of IaaS and PaaS services. Figure 1 shows an overview of the proposal in [6]. A description of how the extension of Brooklyn was carried out to support PaaS providers can be found in [7].

¹ TOSCA (Topology and Orchestration Specification for Cloud Applications) is an OASIS standard for the description of cloud applications and their relationships.

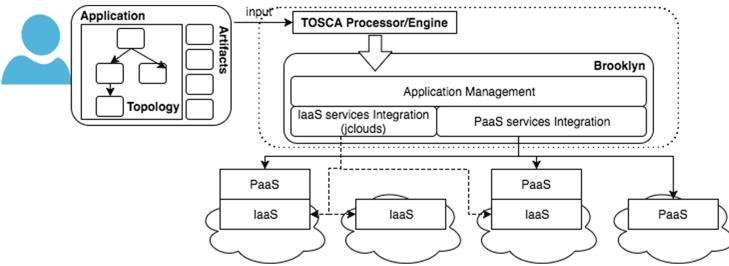


Fig. 1. Trans-Cloud approach

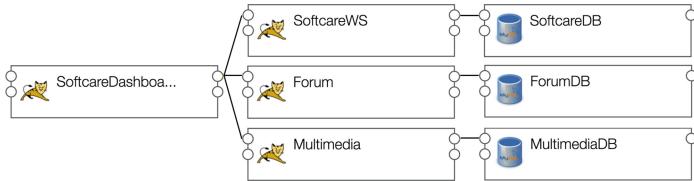


Fig. 2. Brooklyn-TOSCA Softcare's topology

Although the current official release of Brooklyn only handles IaaS services, as can be seen in Fig. 1, Brooklyn's API was extended in [6] with new mechanisms for the management of PaaS services. Our extension takes advantage of the genericity and flexibility of Brooklyn's API, which has the independency between application descriptions and cloud services used in their operation as one of its goals. Initially, CloudFoundry-based platforms, like Pivotal Web Services, IBM Bluemix, etc., were integrated, to prototype the PaaS support, and to allow components to be deployed using both IaaS and PaaS.

As we will see in the following section, the trans-cloud approach provides a set of useful basic mechanisms to build an agnostic algorithm to reach the migration of application's component. Application components are specified in an agnostic way and their control, with operations like stopping, restarting, starting in a new location, can be delegated to the aforementioned unified API, which handles any application component independently of the kind of service where it will be running, either IaaS and PaaS.

2.2 The Softcare Case Study

Softcare is an application for the social inclusion of elderly people and for the management of their medical problems. The application, developed by originally developed at Atos [4], is a cloud-based clinical, educational, and social application, based on state-of-the-art technology.

As depicted in Fig. 2, the application is composed of seven modules: four web modules over respective Tomcat servers (note the Tomcat icons), and three MySQL databases (note the database icons). The **SoftcareDashboard**

Listing 1.1. Softcare's TOSCA description

```

1 tosca_definitions_version: tosca_simple_yaml_1_0_0_wd03
2 ...
3 topology_template:
4   node_templates:
5     SoftcareDashboard:
6       type: org.apache.brooklyn.entity.webapp.tomcat.TomcatServer
7       ...
8     requirements:
9       - endpoint_configuration:
10         node: SoftcareWS
11         ...
12       - endpoint_configuration:
13         node: Forum
14         ...
15       - endpoint_configuration:
16         node: Multimedia
17         ...
18     SoftcareWS:
19       type: org.apache.brooklyn.entity.webapp.tomcat.TomcatServer
20       ...
21     requirements:
22       - endpoint_configuration:
23         node: SoftcareDB
24         ...
25     SoftcareDB:
26       type: org.apache.brooklyn.entity.database.mysql.MySqlNode
27       ...
28     ...
29   groups:
30     add_compute_locations:
31       members: [SoftcareDB, ForumDB, MultimediaDB, Forum]
32     policies:
33       - brooklyn.location: aws-ec2:eu-west-1
34     add_web_locations:
35       members: [SoftcareDashboard, SoftcareWS, Multimedia]
36     policies:
37       - brooklyn.location: softlayer:lon02

```

component provides the main graphical user interface, and depends on the **Forum**, **Multimedia** and **SoftcareWS** modules. **Forum** adds a forum service to the web platform, **Multimedia** is responsible for managing the offered multimedia content, and **SoftcareWS** contains the application's business logic. The databases **ForumDB**, **MultimediaDB** and **SoftcareDB** store, respectively, the forum's messages, the multimedia content, and the rest of the application's data.

Given a TOSCA YAML description with the appropriate location specification, the Brooklyn system will create suitable entities to manipulate each module. Listing 1.1 shows a Softcare's TOSCA YAML topology schema, where just some elements are described to illustrate the agnostic-based application description. We can see how this mechanism allows the separation between topology description and the providers specification. In lines 29–37, we see that the components are deployed on AWS (Ireland's cluster) and SoftLayer (London's cluster), both of them IaaS services.

If we used the TOSCA YAML in Listing 1.2 instead, the components that were targeting SoftLayer would now be deployed on Pivotal (PaaS). The only change between these YAMLS is the target location for some of the modules, with no modification on the original topology or the description of the components.

Listing 1.2. Adding new locations to web modules

```

1 tosca_definitions_version: tosca-simple_yaml_1_0_0_wd03
2 ...
3 groups:
4   add_compute_locations:
5     members: [SoftcareDB, ForumDB, MultimediaDB, Forum]
6     policies:
7       - brooklyn.location: aws-ec2:eu-west-1
8   add_web_locations:
9     members: [SoftcareDashboard, SoftcareWS, Multimedia]
10    policies:
11      - brooklyn.location: pivotal-ws

```

3 Migration Algorithm

Our migration algorithm, specified in Algorithm 1, is completely agnostic, it is just a process orchestrator. Given the abstractions provided by the trans-cloud management tool, and the way application, components and their relations are modeled and handled by it, the migration algorithm is reduced to a minimum. Thus, the management of each concrete application component and bound cloud resource can be delegated to a generic API which hides the complexity of managing different cloud providers. For example, the `stop(component)` operation is in charge of stopping a component regardless of where it is running, either on an IaaS or PaaS service, and the resources it is using.

The migration algorithm is in charge of providing a thread-based parallelized operational plan for the migration of each stateless component to the specified target locations. To minimize waiting times, the algorithm follows a distributed approach. Decisions are made on a component basis, so that each component checks its own dependencies and moves forward in its specific task as soon as possible. The algorithm simultaneously operates on all specified components by following a multithreaded strategy, controlled by the application of a set of rules to detect dependencies between component and cloud resources, and then ensuring the integrity of the application during the migration process.

During the execution of the algorithm, we differentiate two phases: the *stop* phase, where all necessary components—components to be migrated and all other components depending on them—are stopped and cloud resources are released, and the *start* phase, where affected components are re-deployed in desired locations, stopped components are re-started, and connections are re-established. To maximize concurrency during these phases, actions on components are bundled on tasks, namely, `STOP_TASK` (lines 16–28) and `START_TASK` (lines 30–45). For example, a task for stopping a component will contain the necessary operations to stop the component and release its resources. Once a set of tasks have been generated, they will be executed in parallel using the `parallelExecution()` operation of the `executor` object.

The operation `MIGRATE(a,cls)` takes as inputs an in-memory representation of an application’s topology, `a`, and a map `cls` that associates to each of the components of the application to be migrated its target location. The set of keys of this map (`cls.keys()`) represents the components to be migrated.

Algorithm 1. Migration Algorithm

```

1 Input a: application
2 Input cls: component-location map to migrate
3
4 procedure MIGRATE(a, cls)
5   stopTasks = empty list of tasks
6   for c in findComponentsWithNoDeps(a, cls.keys())
7     stopTasks.add(STOP_TASK(a, c, cls))
8   executor(stopTasks).parallelExecution()
9
10  startTasks = empty list of tasks
11  for c in cls.keys()
12    startTasks.add(START_TASK(a, c, cls))
13  executor(startTasks).parallelExecution()
14 end
15
16 function STOP_TASK(a, component, cls): Task
17  return new Task() {
18    stopParentTasks = empty list of tasks
19    for c in parents(a, component)
20      stopParentTasks.add(STOP_TASK(a, c))
21    executor(stopParentTasks).parallelExecution()
22
23    if component in cls.keys()
24      stopAndReleaseResources(component)
25    else
26      stop(component)
27  }
28 end
29
30 function START_TASK(a, component, cls): Task
31  return new Task() {
32    if isReadyToStart(component) {
33      if component in cls.keys()
34        start(component, cls.get(component))
35      else
36        re-start(component)
37      re-establishRelations(component, a)
38
39      startParentTasks = empty list of tasks
40      for c in parents(a, component)
41        startParentTasks.add(START_TASK(a, c, cls))
42      executor(startParentTasks).parallelExecution()
43    }
44  }
45 end

```

Before migrating a component, it is necessary to stop all its input dependencies. For example, if we wanted to migrate a back-end server, which is being used by a set of front-end components, we should ensure that all the components connected to the back-end have been stopped beforehand. The stop phase of the algorithm proceeds recursively on the ancestors of each component of the application (`parents(a, component)`), starting with the set of components with no dependencies in the set of components to stop (`findComponentsWithNoDeps(a, cls)` in line 6), which results in a top-down strategy. Given stop tasks for each of the components in the hierarchy, the concurrent executor will launch each of them in parallel. The stopping task on a component concludes with the invocation of either

the `stopAndReleaseResources(component)` (line 24) or the `stop(component)` (line 26) operations, depending on whether the component in question is one of the components to be migrated or not. These two operations belongs to the trans-cloud API, and will take care of the specificities of the providers and services used. The only difference between them relies on the releasing of the resources used. Those components to be migrated will have new resources bound, and the `stopAndReleaseResources` operation is used on them.

Once all necessary components have been stopped, the target components have to be re-deployed in the new locations. Then, all stopped components, those migrated and their ancestors, must also be either started or re-started. In this phase, if a component has all its dependencies fulfilled (checked by the `isReadyToStart(component)` operation, line 32), the following stand-up operations are applied to it: (1) start/re-start, (2) re-establishment of its connections, and, finally, (3) re-start of its ancestors. The recursive invocation to the ancestors re-starting results in a bottom-up traversing strategy on the dependency structure, which means that a component will only be re-started after all its input dependencies are satisfied. This procedure ensures the topology's integrity and avoid unexpected behavior during the migration. If a specific component has not migrated, it just need to be re-started using the services already bound. If it is one to be migrated, it will be started on the new target location. In both cases, the task concludes re-establishing the relations of the component. Again, all these operations are performed with invocations to the trans-cloud API.

4 The Tool in Practice

In this section, we briefly illustrate how our trans-cloud approach can be used to deploy the application's components on different services simultaneously, and how the migration algorithm can be applied to move some of these components to the specified target locations. We use the Softcare case study introduced in Sect. 2.2 to illustrate the tool and its use. The implementation and the rest of the documentation are available at github, from <https://github.com/scenic-uma/brooklyn-dist/tree/trans-cloud-app-migration>.

4.1 Application Deployment

Our extension of Brooklyn adds to Brooklyn the required mechanisms to support the deployment of components on PaaS vendors, following the trans-cloud concepts. We focus here on the use of Brooklyn-TOSCA and how, thanks to the Brooklyn's plugin, our proposal supports TOSCA-based topology descriptions to be managed by the extended Brooklyn.

In order to deploy an application using our trans-cloud tool, first we need to provide a TOSCA-based topology specification, with all the details on the application topology, its components and how they are related, their properties, configurations, etc. Then, since the trans-cloud mechanisms have been totally integrated under the API in the customized Brooklyn version, we may use the

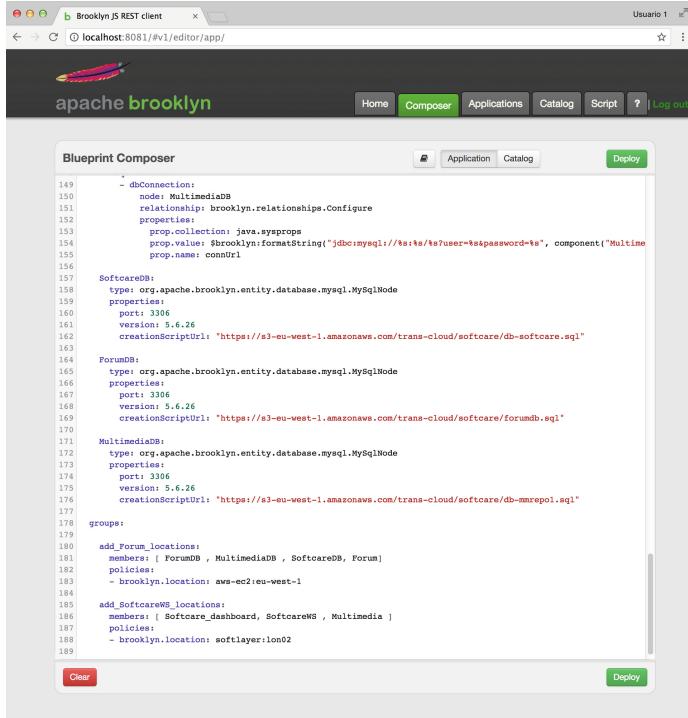


Fig. 3. Deployment of the Softcare application using its TOSCA specification

default Brooklyn Web Console² to deploy our applications using their TOSCA YAML descriptions. For example, given the complete TOSCA YAML description of the case study presented in Sect. 2.2 (an excerpt with its structure is given in Listing 1.1), we may use Brooklyn’s Web Console to deploy the Softcare application on AWS EC2 and SoftLayer as shows the snapshot in Fig. 3.

Once the TOSCA YAML is loaded in Brooklyn’s Blueprint Composer, we may click on the `deploy` button. If so, the topology is processed by the extended Brooklyn (by means of Brooklyn-TOSCA) and the application is deployed using the selected cloud services and resources, AWS EC2 Oregon’s Cluster and Softlayer London’s Cluster.

4.2 Migration Request

As explained in previous sections, the extended version of Brooklyn provides support for the component-wise migration of applications. Precisely, the migration may operate on as many stateless components of an application as desired.

² Deployment Official documentation <https://brooklyn.apache.org/v/latest/start/blueprints.html>.

Although the migration algorithm was developed as an independent orchestrator, since it is included in the customized Brooklyn, we may now apply the migration process to any application which has been deployed (and is therefore being managed) by Brooklyn. Moreover, by the agnosticity of our TOSCA-based application descriptions, we may migrate then without any changes on the application topologies or their descriptions.

Continuing with our case study, and assuming that the Software application has previously been deployed as indicated in the previous section, we explain here how the application can be migrated. Precisely, we show how some of its stateless components, namely **SoftcareDashboard** and **Forum**, are moved to different services. To perform the component-wise migration of an application we must just provide the target locations of the components to be migrated. We could, of course, change the location of all the components of an application. Here we migrate just two of them: **SoftcareDashboard** is moved from SoftLayer's datacenter in London to Amazon EC2, and **Forum** is moved from Amazon EC2 to Pivotal Web Services. Notice that whilst the first migration involves two IaaS providers, the **Forum** component is moved from IaaS to PaaS.

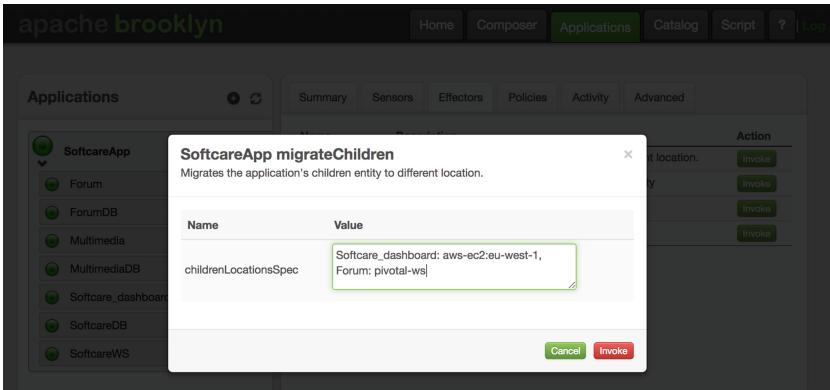


Fig. 4. Migration process launch

Thus, the algorithm just requires a map associating the components to be migrated to the target locations. These locations are selected from a catalog of available datacenters. Figure 4 shows a snapshot of the Brooklyn Web Console with the pop-up window to specify the migration to execute. We can see on the left a tree with the Softcare components running as deployed in the cloud as deployed in the previous section. At the center of the figure we can see the pop-up window with the invocation of the migration process. As value the migration task receives the map configuration with the desired location targets.

Once the map with the target locations has been provided, the migration algorithm is executed by clicking on the **Invoke** button. The algorithm is applied according to its specification in Sect. 3. First, **Forum** is selected to be stopped. The

stop operation on **SoftcareDashboard** is suspended, since it is a direct ancestor of **Forum**. Thus, the algorithm first tries to stop the **Forum** component, but since its ancestors must be stopped before, the **SoftcareDashboard** is then stopped. Once **SoftcareDashboard** is not running, **Forum** is stopped. Because both **SoftcareDashboard** and **Forum** are to be migrated, every bound cloud resources are also released. Then, the re-start process begins. The **SoftcareDashboard** component cannot start directly, because one of its dependencies (**Forum**) is not ready. Hence, **Forum** is started first on **pivotal-ws**. Finally, once **Forum** has been re-deployed and all its relations have been re-established, **SoftcareDashboard** is deployed on AWS. After each start and re-start operation, all necessary relations are re-established to maintain the expected application behavior.

5 Conclusions

An agnostic algorithm to orchestrate the migration process for the stateless component of applications in cloud environments have been proposed. The algorithm allows the parallel migration of several components of an application, and it only needs the mapping of components to be migrated to new locations, independently of the service level (IaaS or PaaS) used both in the source and the target provider. This is possible because the algorithm is based on trans-cloud mechanisms also provided as part of this work. Thus, the proposed algorithm is vendor, technology and service-level agnostic. The migration process is fully automated, and is available on an extended version of Apache Brooklyn. The only required external intervention to carry out the migration is just a migration request to initialize the process.

As future work, we plan to extend our algorithm in several ways, in particular to optimize downtimes. The current version of the algorithm is based on first stopping the components to be migrated, and then starting or re-starting them. To minimize the downtimes we will re-engineer the definitions of lifecycles in Brooklyn to allow the deployment and starting of migrated components before stopping them.

Acknowledgements. This work has been partially supported by Spanish MINECO/FEDER projects TIN2014-52034-R and TIN2015-67083-R; and Univ. Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Androcec, D., Vrcek, N., Kungas, P.: Service-level interoperability issues of platform as a service. In: World Congress on Services (SERVICES), pp. 349–356. IEEE (2015)
2. Binz, T., Leymann, F., Schumm, D.: CMotion: a framework for migration of applications into and between clouds. In: International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–4. IEEE (2011)
3. Boyer, F., Gruber, O., Pous, D.: Robust reconfigurations of component assemblies. In: International Conference on Software Engineering (ICSE), pp. 13–22 (2013)

4. Brogi, A., Carrasco, J., Cubo, J., Nitto, E.D., Durán, F., Fazzolari, M., Ibrahim, A., Pimentel, E., Soldani, J., Wang, P., D'Andria, F.: Adaptive management of applications across multiple clouds: the SeaClouds approach. *CLEI Electron. J.* **18**(1) (2015)
5. Brogi, A., et al.: SeaClouds: a European project on seamless management of multi-cloud applications. *ACM SIGSOFT SEN* **39**(1), 1–4 (2014)
6. Carrasco, J., Cubo, J., Durán, F., Pimentel, E.: Bidimensional cross-cloud management with TOSCA and Brooklyn. In: 9th IEEE International Conference on Cloud Computing (CLOUD), pp. 951–955 (2016)
7. Carrasco, J., Durán, F., Pimentel, E.: Component-wise application migration in bidimensional cross-cloud environments. In: CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, 24–26 April 2017, pp. 259–269 (2017)
8. Durán, F., Salaün, G.: Robust and reliable reconfiguration of cloud applications. *J. of Systems and Software* **122**, 524–537 (2016)
9. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. *Softw. Pract. Exper.* **44**(3), 369–390 (2014)
10. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. *IEEE Trans. Cloud Comput.* **1**(2), 142–157 (2013)
11. Kritikos, K., Plexousakis, D.: Multi-cloud application design through cloud service composition. In: 8th International Conference on Cloud Computing (CLOUD), pp. 686–693. IEEE (2015)
12. Moustafa, A., Zhang, M., Bai, Q.: Trustworthy stigmergic service composition and adaptation in decentralized environments. *IEEE Trans. Serv. Comput.* **9**(2), 317–329 (2016)
13. Paraiso, F., Haderer, N., Merle, P., Rouvoy, R., Seinturier, L.: A federated multi-cloud PaaS infrastructure. In: Chang, R. (ed.) 5th International Conference on Cloud Computing (CLOUD), pp. 392–399. IEEE (2012)
14. Youseff, L., Butrico, M., Silva, D.D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (GCE), pp. 1–10 (2008)
15. Zeginis, D., D'Andria, F., Bocconi, S., Cruz, J.G., Martin, O.C., Gouvas, P., Ledakis, G., Tarabanis, K.A.: A user-centric multi-PaaS application management solution for hybrid multi-cloud scenarios. *Scalable Comput. Pract. Exp.* **14**(1), 17–32 (2013)
16. Zhao, J.-F., Zhou, J.-T.: Strategies and methods for cloud migration. *Int. J. Autom. Comput.* **11**(2), 143–152 (2014)



Verification of the Consistency of Time-Aware Cyber-Physical Processes

Imen Graja¹(✉), Slim Kallel¹, Nawal Guermouche^{2,3}, and Ahmed Hadj Kacem¹

¹ ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
imen.graja@redcad.org

² CNRS, LAAS, 7 avenue du colonel Roche, 31400 Toulouse, France

³ Univ de Toulouse, INSA, LAAS, 31400 Toulouse, France

Abstract. Cyber-physical systems (CPS) represent an emerging type of distributed systems that integrate a multitude of physical elements and software applications into large networks of interconnected components. Ensuring that such systems meet their timing requirements is essential, especially with time-sensitive applications. To deal with this, suitable ways to specify and verify distributed CPS applications including their timing requirements are needed. Current CPS modeling solutions specify CPS as inter-organizational processes using existing process modeling languages. However, the existing process modeling languages mostly focus on web-based workflow and are not directly compatible with CPS. Modeling processes in CPS requires the consideration of cyber elements, physical elements, and their non-functional properties such as time-related and physical properties. Given an inter-organizational CPS processes model with considering structural and non-functional properties, implicit conflicts may arise. To deal with this issue, we propose an approach for modeling and verifying inter-organizational cyber-physical processes associated with temporal properties. To do that, we provide an extended version of BPMN that supports CPS concepts and properties. Then, we define a set of transformation rules to automatically transform the inter-organizational processes model into a constraint satisfaction model. Thereafter, we analyze the generated model to check its consistency.

Keywords: Cyber-physical systems · Temporal properties
Inter-organizational processes · Constraint satisfaction problem

1 Introduction

Cyber-physical systems (CPS) enable the development of complex real-world applications through the integration of computation, communication, control, and physical activities. Embedded computers and networks monitor and control physical functionalities. These functionalities when executed affect the physical world and computations. The design of such systems, therefore, rise new design challenges such as the composition of the cyber and physical functionalities [1].

Solutions for this challenge have been proposed to represent the software and hardware functionalities of CPS in the form of interoperable services [2].

A CPS service can be either a cyber or physical service. Cyber-computational resources, which reside either in static or dynamic host computers in CPS network, provide cyber services. Physical devices (sensors or actuators) provide physical services that monitor or make changes in the real-world environment. Hence, CPS is more complex than web-based applications [3] that only consider software services. In fact, the services offered by cyber-physical resources and computational resources differ in their non-functional properties. The non-functional properties characterize the abstract behavior of CPS and define restrictions on their real-time execution [4]. Since CPS can be applied in a variety of domains - such as medical, transportation, smart home, etc. - we differentiate between two types of non-functional properties; the domain-independent and domain-dependent properties. The domain-independent properties represent general desirable features of different domains, such as the execution time and the cost of executed functionality. The domain-dependent properties, on the other hand, stem from peculiar features of the specified domain and regard its components functionalities. For example, the dimensions of space are important characteristics for many transport systems.

Time-related properties are domain-independent constraints on the CPS behavior over time. They are crucial for the correct functioning of the CPS. In addition, time-related properties can be affected by some physical properties. For example, the velocity can affect the arrival time of a flying drone. The physical properties are the constraints on the environment, physical entities, devices or their behavior in CPS. In fact, the specification of physical properties is related to the application domain of the system. To resume, modeling cyber-physical processes requires capturing important particular CPS aspects such as cyber elements, physical elements and their non-functional properties.

The characteristics of the CPS processes and their associated non-functional properties are arguably more difficult to capture in business process modeling [5,6]. To overcome such limitation, in our previous work, we proposed *BPMN4CPS* [7,8]; a CPS-aware BPMN 2.0 extension with CPS aspects. Our aim is to allow designers to accurately and efficiently model CPS processes. We introduced additional concepts to represent the process logic, different types of activities, CPS resources, real-world elements, time related properties and physical properties. Given a set of inter-organizational CPS processes modeled using *BPMN4CPS*, it is important to set up a verification technique for checking the consistency of the model against the specified properties, i.e., the constraints of each process comply with the different constraints of the different processes.

The modeling of inter-organizational processes is a complex and error-prone step that introduces inconsistencies or errors to the processes, e.g., “dead-lock”. In fact, many approaches have addressed this problem and have proposed some solutions for the verification of the structural correctness of the processes [9] and the satisfaction of the time-related properties [10–12]. However, as described before, CPS activities are constrained by physical restrictions. These

physical properties, which affect greatly the cyber-physical behavior, have not been treated yet, when verifying such processes. In this context, when studying the possible impact of the specified structural, time-related and physical properties on the inter-organizational CPS processes, we found that implicit conflicts may arise that makes the model inconsistent.

To address these issues, we define a novel verification approach based on a constraint satisfaction model to enable the verification of inter-organizational CPS processes. In order to achieve that, we propose a set of transformation rules for the mapping of BPMN4CPS into a constraint satisfaction model.

In this paper, we begin in Sect. 2 by presenting a review of related literature. We introduce our motivating example in Sect. 3. Then, Sect. 4 presents BPMN4CPS. Next, in Sect. 5, we propose our verification approach. In Sect. 6, we describe our BPMN4CPS tool, and we illustrate its use to model and verify the CPS processes. Finally, Sect. 7 concludes the paper.

2 Related Work

Models play an essential role in the design of cyber-physical systems because they allow a designer to analyze, verify, and detect defects of the system early and efficiently. To address this need, researchers have proposed several verification approaches. Unlike the existing verified models, the BPMN4CPS allows specifying a very rich set of structural, time-related, and physical properties.

In the service oriented computing research field, the authors, in [3, 13], describe the CPS functionalities as services to facilitate the composition task. The authors specify the composition of the CPS services as a CPS workflow pattern, which is a sequence of actions that move the system from the initial state to the desired goal. Since physical properties may impact the operation of the services and affect their availability and functionality, the proposal in [14, 15] effectively support the specification of the physical properties. The above approaches consider the structural, physical and some time-related properties. Recently, authors use BPMN to model CPS processes. In [6], the authors extend BPMN to model resources of the real-world entities that perform CPS activities and specify resource-related non-functional properties such as the performance and reliability for each component. In order to model self-adaptive workflows for cyber-physical systems. Seiger et al, in [16, 17], present a process-based framework in which the abstract system behavior is modeled using BPMN. Furthermore, to consider real-world context situations, in [18], the authors introduce SmartPM which is a framework for automated process adaptation in case of unanticipated exceptions. In fact, in the above approaches, the designer uses the BPMN to define the process control flows among a set of tasks. These approaches show the strong need for modeling complex collaborative behavior of distributed CPS and expressing both sets of structural properties and complex flows of data (event-based) within high-level CPS processes. Despite the fact that these models consider structural, physical and some time-related properties, however, they do not take into account a detailed set of richer time-related properties that

can affect the composition of cyber and physical functionalities, as well. Most importantly, these approaches do not apply any verification guidelines for the analysis of the correctness of the workflow. In short, the use of BPMN was only for CPS modeling needs, so verifying the correctness of such models have not been investigated.

Some modeling approaches focus on the specification of CPS using mathematical languages to be checked. To this end, formal verification of CPS, such as a model checking technique, is widely used with the aim of verifying different goals. The authors, in [19], propose a probabilistic approach to formally describe and analyze the reliability and cost-related properties of a composed set of services in IoT. Formal techniques have also been applied to verify the deadlock, livelock [2], reachability and safety [20–22] of a cyber-physical model. In particular, in service-oriented or business systems, works have shown the importance of the temporal consistency analysis, by applying either static (i.e., at-design-time) [11, 23] or dynamic (at-run-time) [24] verification. These temporal verification approaches check whether a temporal violation occurs or not, based on formal methods. Nevertheless, these approaches are limited to discover only some time-related inconsistencies, and they do not consider other eventual physical-related conflicts that can arise when concurrent behavioral cyber-physical processes interact together (i.e., inter-organizational processes). In general, the model checker examines all possible system scenarios to identify if a given model truly satisfies certain properties. However, they may not be capable to prove physical property and to find their possible values. Hence, this technique is insufficient to handle our problem that consists of verifying and finding the possible combinations of the different attribute values [25]. This problem can be solved through using the constraint satisfaction problems. A constraint satisfaction problem is a mathematical specification to represent and solve a combinatorial problem. This technique has been used for scheduling problems [26], service selection problems [27], etc. and it can address our verification requirements.

3 Motivating Example

When a hurricane, earthquake, or other natural disaster strikes, lives depend on the response, delivery of supplies and assistance on time. The disaster recovery systems is a communication between the search system and the delivery system. The designer specifies such behavior using two inter-organizational cyber-physical processes, as shown in Fig. 1. First, the search system encompasses a set of sensor drones, installed car cameras and outdoor cameras that monitor the environment. The sensor drones are equipped with wireless routers and antennas to set up a WiFi network in the damaged area. Other water-level sensors installed in the scene, collect information about the environment. The cyber part of such system allows the analysis and the communication with the delivery system. Second, the delivery system encompasses a set of delivery drones and self driving vehicles that carry supplies to the damaged location. The cyber part of

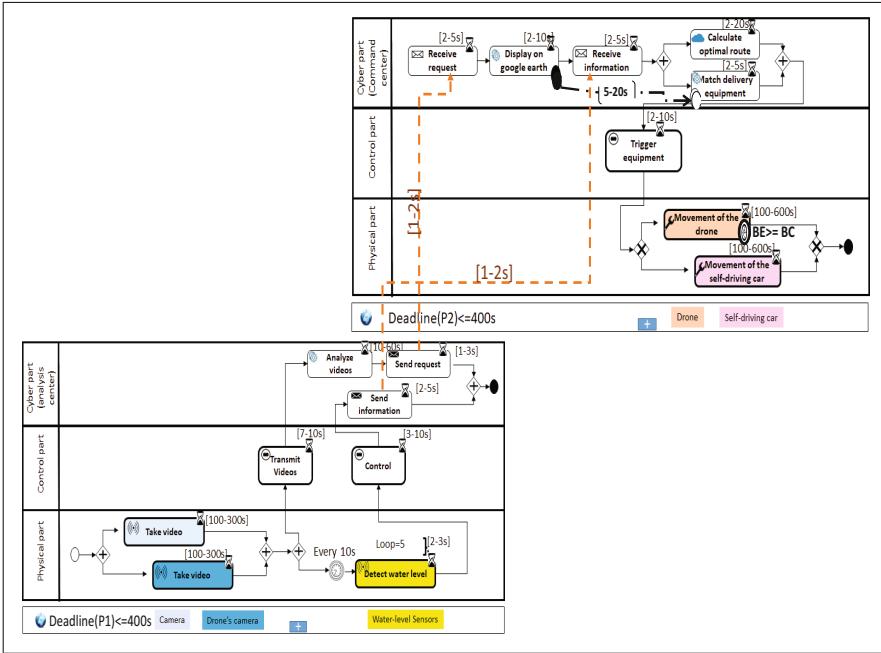


Fig. 1. Processes of the disaster recovery systems

such system allows displaying incoming requests on a map using Google earth, and compute an optimal mission plan.

Before executing such systems, using BPMN4CPS, the designer models their collaborative behavior, with considering some time-related and physical requirements. The aim is to verify the correctness of the behavior and whether or not the execution will satisfy the specified properties. The associated time related properties are as follows:

- The duration of each activity in the process, such as the duration of carrying items (i.e., *movement-of-the-drone*) activity is between 100 to 600 s.
- The *detect-water-level* activity has a recurrent behavior that must be repeated every 10 s and no more than 5 times. The aim is to check if the level is increasing.
- The start time of the *match-delivery-equipment* activity must be between 5 to 20 s from the finish time of the *display-on-google-earth* activity.
- The communication is synchronous and takes 1 to 2 s.
- The deadline of the collaborative behavior is 400 s.

In addition, the designer specifies as physical properties; (1) a constraint on the drones battery consumption that must ensure its return to the depot after delivering the items, and (2) a global physical property that represents a relationship between two physical attributes in different processes. The first property is specified as a must happen at finish time property. The second property indicates

that the drone has to be far away from the water covered region. So, the position of the executed water-level sensors must be different from the position of the drone: $P_{drone} \neq P_{WaterS}$. We note that the position is $P(x_i, y_i)$ or a whole area, according to the designer specification. Next, we will give a brief overview of the BPMN4CPS.

4 BPMN4CPS: A BPMN Extension for Modeling Time-Aware CPS Processes

In order to enable the modeling of CPS processes, we proposed, in [7], an extended version of BPMN. The proposed extensions introduce new key concepts related to the CPS functional and non-functional properties. We also supply the modeler with the possibility to visually specify these different CPS aspects and non-functional properties [8]. In what follows, we present the CPS concepts supported by BPMN4CPS, and we describe the newly improved and refined concepts.

4.1 Modeling CPS Aspects

For modeling CPS, we introduce additional concepts, with the purpose of specifying *the logic of the CPS processes, CPS activity types, resource roles and real-world elements*.

Logic of CPS processes: the CPS application runs on different types of entities, which their behavior can be seen as three distinct participants. The created process model is later used to derive the code that will be executed by the CPS. For such reasons, supporting the deployment of the process requires to split the process model into a cyber, physical and control parts. These parts of the process have to be separated and handled differently. Hence, we proposed the three processes logic, that allows the designer to model the CPS behavior as a collaboration of three process participants. We also introduce a new extension that forces the modeler to specify the cyber-physical system as a single process structured in three parts.

CPS activity types: there are three types of tasks in CPS: the cyber, manual and physical task. The cyber and the physical tasks have different sub-types of tasks. The cyber task can be a web-based, cloud-based or embedded task, while the physical task can be either a sensor's task or actuator's task.

CPS resources: in our previous work, we proposed the device performer only as an extension, while in CPS both cyber and physical resources are equally important. Therefore, in CPS model, the performer can be *Computing Performer* for cyber tasks or *Device Performer* for physical tasks.

Real-world environment and physical entities: this extension has been proposed to support the specification of the real-world environment (RWE), as an empty pool, which is a participant containing the different physical entities that can be *affected* or *monitored* by a physical task.

4.2 Modeling Time-Related and Physical Properties

In [8], we proposed an extension of BPMN4CPS to support time-related and physical properties. First, we consider the following time-related properties:

- Start and finish time of an activity.
- Duration of an activity as the minimum and maximum time taken to be executed.
- Recurrent activity or set of activities, that is restricted by a number of execution, and either an interval of time between two starts or an interval of time in which the behavior can be executed.
- Temporal-dependency between the start/finish and start/finish of two different activities.
- Communication time between two activities in two different processes.
- Deadline, which is the time taken to execute one process or a set of collaborative processes.

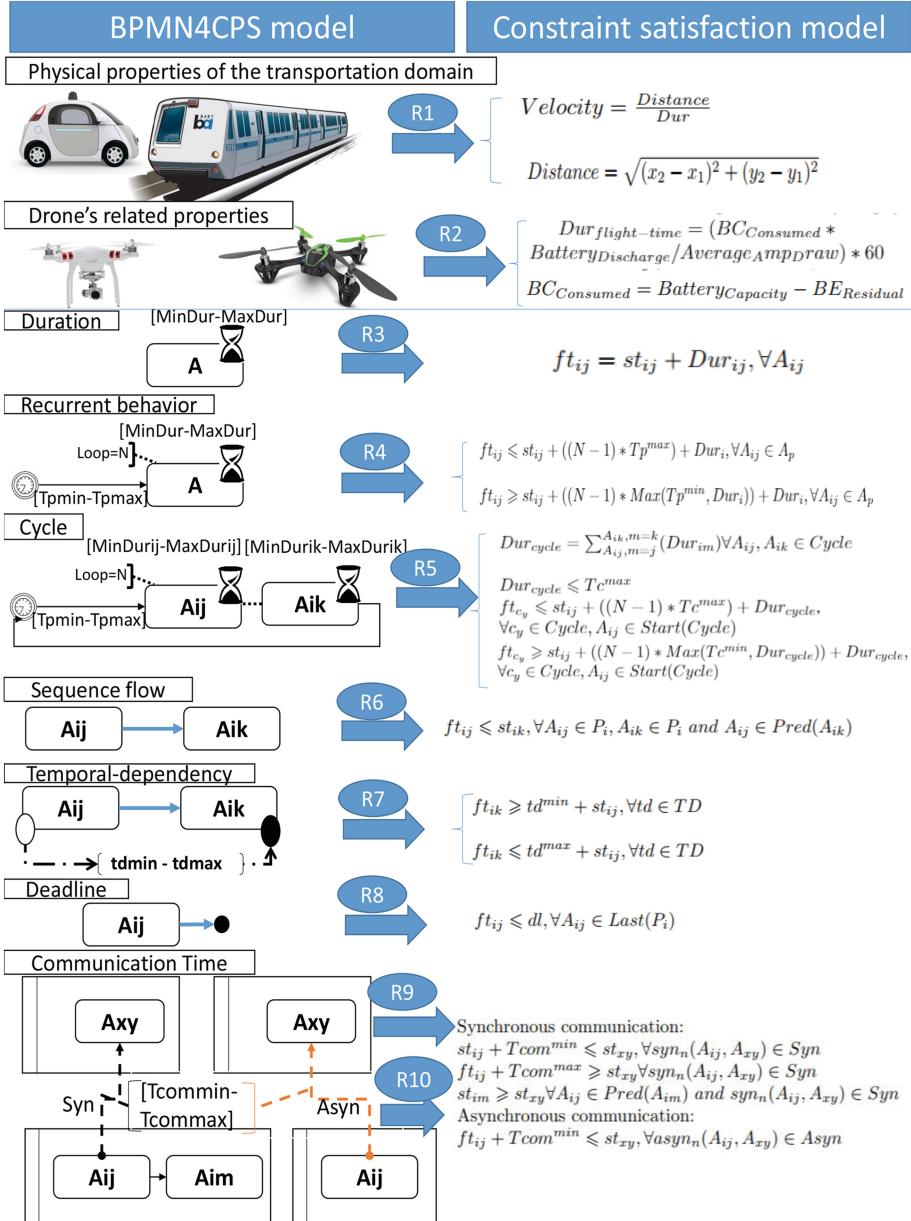
Second, we consider the following physical properties:

- Must happen at the start or finish time of an activity.
- Must happen during the execution of an activity.
- Must happen in a bound time.
- Must happen in an infinite time.
- Global physical properties that allow the specification of constraints on the whole collaborative behavior such as the relationship between two or more physical attributes from different activities and processes.

5 Verification of the Consistency of the CPS Processes

The verification approach we propose relies on a constraint satisfaction model. In order to achieve that, we define a set of mapping rules that transform the BPMN4CPS model into a constraint satisfaction problem (CSP) [28]. We remind you that verification of collaborative CPS processes aims to assert that the different constraints of the involved processes do not give rise to conflicts, i.e., inconsistencies, which can be an obstacle towards their collaboration. The constraint programming is a technique that takes its features from different domains such as the operational research and the artificial Intelligence. This technique helps solving real combinatorial problems. Using the constraint satisfaction problems, we can specify various parameters as variables and express the dependencies between collections of variables as a set of constraints. The goal of this technique is to define the possible values of all the decision variables, while all the specified constraints are satisfied.

The mapping process starts first by transforming the structural, time-related and physical properties of each cyber-physical process into a set of variables and constraints. Then, we map the properties related to the communication and global behavior of the collaborative CPS processes into a set of constraints. This mapping is based on a set of transformation depicted in Fig. 2. Unfortunately,

**Fig. 2.** Transformation rules

in this paper, due to space limitations, we did not introduce all the rules such as those that associate a domain for each variable and those that transform other structural dependencies between activities (e.g., the multi-choice structure).

We note that we consider some assumptions that need to be considered when modeling the systems. First, we propose to consider quantitative attributes. Qualitative physical attributes can be also supported since they can be represented as quantitative attributes based on Boolean metrics. Second, studying the recurrent behavior of a set of activities is a complicated step that requires some restrictions. To do that, we assume that first the cycle does not contain any communicating activities such as the send or receive activity. Finally, the activities in the cycle do not belong to any other cycle. This assumption ensures that the model does not contain overlapping cycles.

5.1 Transformation Rules for a Cyber-Physical Process

In our previous work [8], we proposed a set of constraints for a cyber-physical process only. However, this proposal lacks considering the cycle and the difference between multi-choice and parallel structure. Therefore, in this paper, we extend our previous work to address these issues. To do so, we present the transformation rules that must be applied to each process, in order to generate the set of needed variables and constraints. These rules are those from R_1 to R_8 given in the Fig. 2. In particular, the R_1 and R_2 allows the transformation of physical properties to a set of constraints. These constraints are related to the application domain. Therefore, we define the physical properties associated with the case study. These properties can be either specified by the designer or they can be automatically extracted from a domain model.

5.2 Transformation Rules for Collaborative CPS Processes

To verify the consistency of the communicating cyber-physical processes, in [29], we proposed a new set of constraints that allow the verification of the structural, time-related, physical and synchronization properties. However, the proposed mapping of the cycle is a very complex process that needs to transform each execution trace of the cycle to a set of variables and dependencies between these variables. In fact, this operation is very hard to achieve especially with very complex processes. In addition, the proposed constraints on the communication time do not differentiate between synchronous and asynchronous communications. Therefore, in this paper, we propose new transformation rules that are efficient to generate better and accurate results as given in Fig. 2. In particular, the rule R_9 allows the specification of the synchronous communication between processes, guarantees that the communication time is satisfied, and specifies that the process remain blocked until the operation completes. Whereas, the rule R_{10} transforms the asynchronous communication that is a non-blocking communication. Finally, we solve the transformed mathematical set of combinatorial problems to check for consistency.

6 Experimentation: BPMN4CPS Tool

We implement BPMN4CPS as a plug-in for Eclipse, which is an extension of the BPMN2 Modeler. The implementation starts by extending BPMN to supports

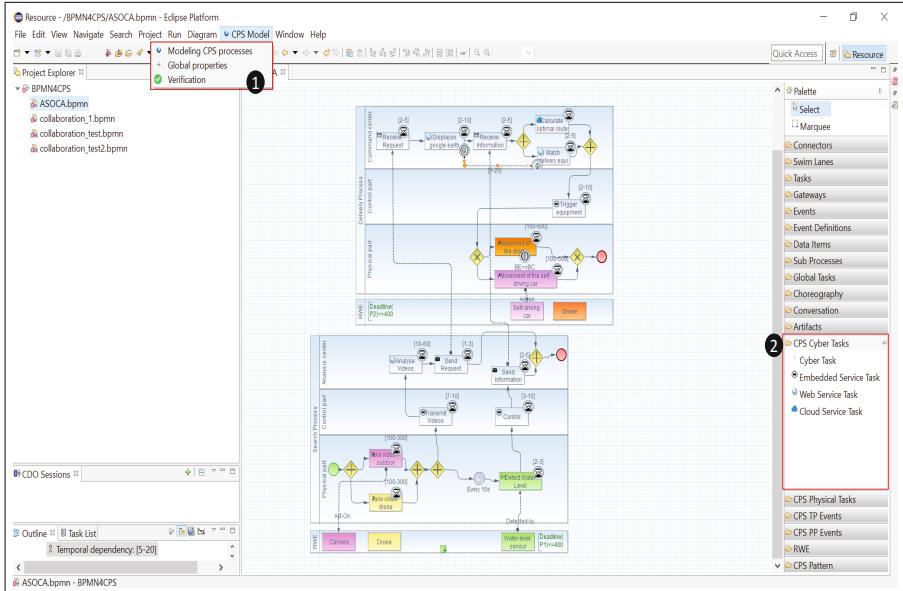


Fig. 3. BPMN4CPS plugin

```
Consistency=TRUE
T = 250
Dur_Take_video = 100st Take_video = 0ft Take_video = 100
Dur_Detect_Water_level = 2st Detect_Water_level = 100ft Detect_Water_level = 142Maximum = 101loopT = 1021 = 4022 = 40
Dur_Transmit_videos = 7st Transmit_videos = 100ft Transmit_videos = 107Dur Analyse_videos = 10st Analyse_videos = 107ft Analyse_videos =
```

Fig. 4. Result of the consistency verification

CPS aspects. Second, we extend BPMN4CPS to support time-related and physical properties. Hence, the plug-in allows designers to model the CPS processes and their collaborative behavior constrained by time-related and physical properties [30]. Third, to verify the consistency of the model, we add another feature to the BPMN4CPS plug-in that allows the automatic mapping of the processes model to a constraint satisfaction problem written in Java. To do that, we integrate the Choco solver [28] into the BPMN4CPS plug-in. In Fig. 3, we show a screen-shot of BPMN4CPS. Our tool allows the designer to model CPS processes, specify global properties and verify the model, as depicted by label ① in Fig. 3. We also extend the palette to include the different CPS concepts and properties, such as the cyber tasks, as shown by label ② in Fig. 3.

Based on the quality characteristics defined in ISO 9126, and the most used criteria and metrics for evaluating methods and tools inspired from [31, 32], we evaluate our approach and the developed BPMN4CPS plug-in. We prove the functionality, usability and effectiveness of our proposed approach. If we go back to the motivating example, the application of the transformation rules allows

us to generate the constraint satisfaction model and automatically analyze the specified model. Based on the generated results as given in Fig. 4, we can prove that the model of the disaster recovery systems is consistent, and it ensures the satisfaction of its specified requirements.

7 Conclusion

This paper studied the verification of the consistency of inter-organizational cyber-physical processes with a focus on their time-related and physical properties. We presented BPMN4CPS, which is an extension of BPMN 2.0 to support CPS aspects and relevant properties, namely, the physical and time-related properties. Next, we described a novel verification approach that checks the consistency of the model. To accomplish the verification, we proposed a set of transformation rules that map the modeled CPS processes into a constraint satisfaction model. The resulting model can be solved in order to verify the consistency of the inter-organizational cyber-physical processes. Finally, we described the BPMN4CPS tool, and we evaluated our approach through an example of disaster recovery systems. We are currently working on BPMN4CPS plug-in and adding a number of additional features, such as support for complex physical properties. In the future, we intend to apply a dynamic (at-run-time) verification to analyze the temporal consistency.

Acknowledgments. This work has been co-funded by a FEDER-FSE 2014-2020 fund of the Region Midi-Pyrénées and the European Union and also by French Government (program: investment for future) in the project: Smart Services for Connected vehiCles-S2C2.

References

1. Huang, J., Bastani, F.B., Yen, I.L., Zhang, W.: A framework for efficient service composition in cyber-physical systems. In: International Symposium on Service Oriented System Engineering, SOSE 2010, pp. 291–298 (2010)
2. Wang, P., Xiang, Y., Zhang, S.H.: Cyber-physical system components composition analysis and formal verification based on service-oriented architecture. In: International Conference on e-Business Engineering, ICEBE 2012, pp. 327–332. IEEE (2012)
3. Wan, K., Alagar, V.: A resource-centric architecture for service-oriented cyber physical system. In: Park, J.J.J.H., Arabnia, H.R., Kim, C., Shi, W., Gil, J.-M. (eds.) GPC 2013. LNCS, vol. 7861, pp. 686–693. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38027-3_74
4. Lohmann, D., Schroder-Preikschat, W., Spinczyk, O.: Functional and non-functional properties in a family of embedded operating systems. In: International Workshop on Object-Oriented Real-Time Dependable Systems, WORDS 2005, pp. 413–420 (2005)
5. Meyer, S., Sperner, K., Magerkurth, C., Debortoli, S., Thoma, M.: Internet of Things architecture IoT-a project deliverable D2.2 concepts for modelling IoT-aware processes (2012). <http://www.meet-iot.eu/deliverables-IOTA/D2.2.pdf>

6. Bocciarelli, P., D'Ambrogio, A., Giglio, A., Paglia, E.: A BPMN extension for modeling cyber-physical-production-systems in the context of industry 4.0. In: International Conference on Networking, Sensing and Control, ICNSC 2017. IEEE (2017)
7. Graja, I., Kallel, S., Guermouche, N., H. Kacem, A.: BPMN4CPS: a BPMN extension for modeling cyber-physical systems. In: IEEE International Conference on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2016, pp. 152–157 (2016)
8. Graja, I., Kallel, S., Guermouche, N., Kacem, A.H.: Modeling and verification of temporal properties in cyber-physical systems. In: IEEE Consumer Communications & Networking Conference, CCNC 2017 (2017)
9. van der Aalst, W.M., ter Hofstede, A.H.: Verification of workflow task structures: a petri-net-baset approach. *Inf. Syst.* **25**(1), 43–69 (2000)
10. Du, Y., Tan, W., Zhou, M.: Timed compatibility analysis of web service composition: a modular approach based on petri nets. *IEEE Trans. Autom. Sci. Eng.* **11**(2), 594–606 (2014)
11. Han, R., Liu, Y., Wen, L., Wang, J.: Probability timing constraint WF-nets and their application to timing schedulability analysis of workflow management systems. In: World Congress on Computer Science and Information Engineering, CSIE 2009 (2009)
12. Du, Y., Tan, W., Zhou, M.: Timed compatibility analysis of web service composition: a modular approach based on petri nets. *IEEE Trans. Autom. Sci. Eng.* **11**(2), 594–606 (2013)
13. Huang, J., Bastani, F., Yen, I.L., Jeng, J.J.: Toward a smart cyber-physical space: a context-sensitive resource-explicit service model. In: Computer, Software and Applications Conference, COMPSAC 2009, pp. 122–127. IEEE, Seattle (2009)
14. Zhu, W., Zhou, G., Yen, I.L., Bastani, F.: A PT-SOA model for CPS/IoT services. In: International Conference on Web Services, ICWS 2015, New York, pp. 647–654. IEEE, June 2015
15. Zhao, Y., Dong, J., Huang, J., Zhang, Y., Yen, I.L., Bastani, F.: Service life cycle tools and technologies: methods, trends and advances. In: Abstract Service for Cyber Physical Service Composition. Information Science Reference, pp. 303–322 (2012)
16. Seiger, R., Huber, S., Schlegel, T.: Toward an execution system for self-healing workflows in cyber-physical systems. *Softw. Syst. Model.* **17**(2), 551–572 (2016)
17. Seiger, R., Huber, S., Schlegel, T.: PROtEUS: an integrated system for process execution in cyber-physical systems. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) CAiSE 2015. LNBI, vol. 214, pp. 265–280. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19237-6_17
18. Marrella, A., Mecella, M., Halapuu, P., Sardina, S.: Automated process adaptation in cyber-physical domains with the smartPM system. In: IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2015, pp. 59–64, October 2015
19. Li, L., Jin, Z., Li, G., Zheng, L., Wei, Q.: Modeling and analyzing the reliability and cost of service composition in the IoT: a probabilistic approach. In: International Conference on Web Services, ICWS 2012, Honolulu, pp. 584–591 (2012)
20. Arney, D., Pajic, M., Goldman, J.M., Lee, I., Mangharam, R., Sokolsky, O.: Toward patient safety in closed-loop medical device systems. In: International Conference on Cyber-Physical Systems, ICCPS 2010, New York, pp. 139–148. ACM (2010)

21. Li, T., Tan, F., Wang, Q., Bu, L., Cao, J.N., Liu, X.: From offline toward real time: a hybrid systems model checking and CPS codesign approach for medical device plug-and-play collaborations. *IEEE Trans. Parallel Distrib. Syst. (TSMCA)* **25**(3), 642–652 (2014)
22. Banerjee, A., Gupta, S.K.S.: Spatio-temporal hybrid automata for safe cyber-physical systems: a medical case study. In: International Conference on Cyber-Physical Systems, ICCPS 2013, New York, pp. 642–652. ACM (2013)
23. Li, J., Fan, Y., Zhou, M.: Timing constraint workflow nets for workflow analysis. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **33**(2), 179–193 (2003)
24. Liu, X., Wang, D., Yuan, D., Wang, F., Yang, Y.: Throughput based temporal verification for monitoring large batch of parallel processes. In: International Conference on Software and System Process, ICSSP 2014, New York. ACM (2014)
25. Krishna, S.N., Trivedi, A.: Hybrid automata for formal modeling and verification of cyber-physical systems. *J. Indian Inst. Sci.* **93**(3) (2015)
26. Barták, R., Salido, M.A., Rossi, F.: Constraint satisfaction techniques in planning and scheduling. *J. Intell. Manuf.* **21**(1), 5–15 (2010)
27. Guidara, I., Guermouche, N., Chaari, T., Jmaiel, M., Tazi, S.: Time-dependent QoS aware best service combination selection. *Int. J. Web Serv. Res.* **12**(2), 1–25 (2015)
28. Prud'homme, C., Fages, J.G., Lorca, X.: Choco Documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. (2016)
29. Graja, I., Kallel, S., Guermouche, N., Kacem, A.H.: Towards the verification of cyber-physical processes based on time and physical properties. *Int. J. Bus. Syst. Res.* (2017)
30. Graja, I., Mechim, A., Kallel, S., Guermouche, N., Kacem, A.H.: Demonstrating BPMN4CPS: modeling and verification of cyber-physical systems. In: IEEE Consumer Communications & Networking Conference, CCNC 2017 (2017)
31. Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case studies for method and tool evaluation (1995)
32. Sacha, K.: Software engineering: evolution and emerging technologies. In: Evaluation of Software Quality. IOS Press (2005)



Model Checking of Cost-Effective Elasticity Strategies in Cloud Computing

Rawand Guerfel¹(✉), Zohra Sbaï^{1,2}, and Rahma Ben Ayed¹

¹ Université de Tunis El Manar, École Nationale d'Ingénieurs de Tunis,
BP. 37 Le Belvédère, 1002 Tunis, Tunisia

{Rawand.Guerfel,zohra.sbai,rahma.benayed}@enit.rnu.tn

² College of Computer Engineering and Science,
Prince Sattam Bin Abdulaziz University, PO Box 151,
Al-Kharj 11942, Kingdom of Saudi Arabia

Abstract. Cloud computing is a revolution in how computing power is delivered to business. It offers different measured services to clients who require them by writing a simple request. These requests are becoming more and more complex so that services need to be composed to meet them. Additionally, these Cloud composite business services (CCBSs) need to be elastic, i.e. their number should be replicated or reduced according to the number of their user demands. Ensuring these two operations is done according to a well-defined strategy. We are interested in this paper in cost-effective elasticity one. Applying this strategy on CCBSs gives birth to a system that needs to be checked to insure that SLA constraints, such as deadline specified by the user, are not violated. In this paper, we present a formal model using Timed Coloured Petri nets to model, check and compare between these strategies before implementing them in real Cloud.

Keywords: Cloud computing composition · Cloud elasticity
Cost-effective strategy · SLA · Formal model
Timed Coloured Petri Net

1 Introduction

Cloud computing is a service model where different resources and data such as servers, switches, storage, applications and services are accessed over the internet. It is a model that enables ubiquitous on demand access to a shared pool of configurable computing resources which can be rapidly provisioned and released with minimal management effort.

There are basically three layers to the Cloud that are used differently based on what they offer. The first layer is Infrastructure as a Service (IaaS) which offers virtual systems that can be connected using internet. The second layer is Platform as a Service (PaaS) which is a proof model for running applications without the hassle of maintaining the hardware and software infrastructure at

the company. The last layer is Software as a Service(SaaS) which is a delivering way of application as a service. Using this layer, one is not obliged to install and maintain software [13].

Nowadays, user requirements are becoming more and more complex that sometimes, a single Cloud business service cannot meet user requests. It needs to communicate and to be combined with other business services to respond to user demands. In this case, we are treating the composition mechanism in Cloud computing, an already discussed issue in our previous works [3,4].

The challenge with these composite Cloud services is that they should be elastic [6]. Indeed, elasticity is one of the most important characteristic that Cloud computing offers to its users. More precisely, it allows the providers to adapt in term of numbers to the user demands in a transparent way. We distinguish two types of elasticity. The first type is vertical elasticity which is related to the scaling up or down of resources of a specific Cloud service without modifying its number (e.g.: the power or the capacity of servers). The second type is horizontal elasticity which is related to the removing or the adding of Cloud service instances. In our work, we are interested in horizontal elasticity. More precisely, in horizontal elasticity type, when the number of user demands for the CCBS increases, the provider has to replicate as many copies of service instances as this number. Similarly, when the number of user demands decreases, the provider has to delete the unused copies of service instances. Replicating and deleting actions are done according to an elasticity-strategy.

Executing elasticity strategies on composite Cloud business service does not necessarily imply the replication or the deleting of the whole composite service. Indeed, according to some indicators, we can apply these operations on only some of the services involved in the composition. This gives birth to a system composed of different services that interconnect and communicate between each other. Users access should be well organised so that the system will not suffer from some problems such as deadlock, conflit, etc.

Let's also note that when applying and choosing an elasticity strategy, one has to take into account two major factors which are: the deadline parameter specified by the user and the gained cost for the Cloud provider. Indeed, on the one hand, SLA has to be ensured. More particularly, when applying an elasticity strategy, we have to check that the deadline is not violated so the provider does not pay a penalty. On the other hand, we should make sure that the elasticity maximizes the cost gained by the provider. This leads to the use of what is known by "cost-effective elasticity strategy".

It is in this context that our work is oriented. Indeed, we propose to check the validity of the model obtained when executing cost-effective elasticity strategies on composite Cloud business services. This model should ensure the non-violence of the user deadline constraint. To do so, we use formal modelling. More precisely, we use Coloured Petri nets (CPN) [8]. CPN allow us to assign time and data information to each service, so that we can assign cost to each service. Besides, CPN offers us the possibility to differentiate between multiple users. Indeed,

coloured tokens are associated with data that could be used as specific ID for each user.

The reminder of this paper is structured as follows. We start the Sect. 2 with presenting the system model by giving some important definitions. Then, we give a motivation example based on two existing strategies. Then, we move to the Sect. 3 to model the elasticity strategy using CPN tool [14]. Also, in this section, an algorithm of generation of CCBS model in CPN is detailed. To validate this model, we check some properties detailed in Sect. 4. Afterwards, we refer, in Sect. 5, to related works. In Sect. 6, we present conclusions and expose some future works.

2 Cost-Effective Elasticity Strategy of Composite Cloud Business Services

Cloud services should be characterized by one of the most important benefits offered by the Cloud computing which is: elasticity. In fact, by rapid elasticity, the Cloud can dynamically allocate or deallocate resources based on the customer configurations [6]. As we already mentioned, this work focuses on horizontal elasticity.

Horizontal elasticity of composite Cloud business services means adapting the number of this composite service to the number of user demands. Many indicators exist to help the provider know when to apply the elasticity strategy namely: the number of user demands, the maximum/minimum number of active sessions, number of user request per unit of time, etc. In our work, we are interested in elasticity strategies that are based on these two indicators:

- the maximum and minimum number of user demands that each service can hold.
- the cost gained when applying the elasticity strategy.

2.1 System Model

Ensuring the elasticity of composite Cloud business services is an important and necessary step. To do so, many strategies have been proposed. To execute the necessary actions, most of them are based on the number of users accessing the CCBS as an elasticity indicator. However, we are interested in ones that add the cost factor when deciding to process these actions.

In this section, we give some important definitions.

Definition 1 (CBS definition): A CBS_i is defined with the tuple $(name_i, max-thres_i, min-thres_i, Resp-T_i, cost_i, Rep-Cost_i)$ where:

- $name_i$: is the name of CBS_i
- $max-thres_i$: is the maximum threshold of CBS_i
- $min-thres_i$: is the minimum threshold of CBS_i
- $Resp-T_i$: is the response time of CBS_i

- $cost_i$: is the cost of the CBS_i
- $Rep-Cost_i$: is the cost of replication action of the CBS_i

Definition 2 (CCBS definition): A $CCBS_0$ is a combination of n CBS_{oi} ; where $i \in 1..n$; and has the following structure:

$$\begin{aligned} CCBS_0 &= (CBS_{01}, CBS_{02}, \dots, CBS_{0n}) \\ &= (<name_{01}, max-thres_{01}, min-thres_{01}, Resp-T_{01}, cost_{01}, \\ &\quad Rep-Cost_{01}>, \dots, <name_{0n}, max-thres_{0n}, min-thres_{0n}, \\ &\quad Resp-T_{0n}, cost_{0n}, Rep-Cost_{0n}>) \end{aligned}$$

Definition 3 (CCBS's response time): The response time of one $CCBS_i$, noted by $Resp-T_i$, is the sum of the response time of all its CBS s. It is defined as follows:

$$Resp-T_i = \sum_{j=1}^n Resp-T_{ij} ;$$

Where n is the number of CBS s that compose the $CCBS$.

Before accessing these services, a SLA is defined between the user and the provider. In this SLA, users specify some constraints that should be valid such as: the availability, the deadline, the budget, the penalty, etc. Some of these parameters are treated in this paper and detailed in the following definition.

Definition 4 (User requirement): A user requirement is given as follows: $UR=(name, dead, budget, penalty)$; knowing that:

- *Name*: is the name of either the $CCBS$ or the CBS . In fact, the user can access to just one Cloud business service.
- *Dead*: is the maximum time given to the provider to respond to the user requirement.
- *Budget*: is the price offered to the provider if the service is given before Dead.
- *Penalty*: is the penalty to be paid by the provider if the request is given after Dead, the parameter specified by the user.

2.2 Motivation

Let's suppose that we have four Cloud business services CBS_{11} , CBS_{12} , CBS_{13} and CBS_{14} composing the $CCBS_1$.

Each CBS is defined as follows:

$$CBS_{11} = (CBS_{11}, 30, 5, 0.3, 0.2, 0.25)$$

$$CBS_{12} = (CBS_{12}, 35, 5, 0.4, 0.35, 0.45)$$

$$CBS_{13} = (CBS_{13}, 40, 3, 0.5, 0.55, 0.7)$$

$$CBS_{14} = (CBS_{14}, 21, 6, 0.2, 0.15, 0.2)$$

Let's suppose that this CCBS is required by multiple users in the same time. So, an elasticity strategy must be applied.

Let's suppose that at time t1, 20 users want to access to CCBS. Note that the deadline of 2 users are not respected.

Then, at time t2, 20 other users want to access to CCBS. All deadlines are respected but only 15 users are a cost gain for the provider perspective.

Finally, at time t3, 25 users leave.

To make the necessary decision, an elasticity strategy has to be applied. In fact, our work is based on two essential elasticity strategies detailed in the following.

Elasticity strategy 1 [5]:

First of all, the provider checks in every unit of time (e.g. second) the maximum and the minimum threshold of each CBS. If the maximum one is reached, then, he calculates the cost benefit when applying a replication action. If the cost is a gain for the provider perspective, then, a replication action is processed. Else, he waits for more users having a higher budget. Let's note that if the minimum threshold of one CBS is reached, then, a deletion action of this service is automatically done. Indeed, the provider has nothing to lose when executing this action. In this strategy, they supposed that all deadlines specified by users are already checked.

Applying this strategy to the previous scenario gives the following result :

At time t1, accept only 18 users. Here, no service will be replicated.

At time t2, accept only 15 users. In this case, CBS_{11} and CBS_{14} reach their maximum thresholds and have to be replicated to CBS_{21} and CBS_{24} .

At time t3, eliminate 25 users. CBS_{21} and CBS_{24} reach their minimum thresholds and have to be removed and replaced by CBS_{11} and CBS_{14} .

Elasticity strategy 2 [7]:

In this strategy, the provider checks in every unit of time if the deadlines specified by users are respected. If 90% of users have a response time lower than their deadlines, so, accept the other 10% of users whatever the penalty to be paid and do the replication action. Else, if less than 90% have a response time lower than their deadlines, the provider does not accept them and waits for other users that satisfy this condition.

Applying this strategy to the previous scenario gives the following result:

At time t1, accept 20 users. Here, no service will be replicated.

At time t2, accept all the 20 users. In this case, CBS_{11} , CBS_{12} and CBS_{14} reach their maximum thresholds and have to be replicated to CBS_{21} , CBS_{22} and CBS_{24} .

At time t3, eliminate 25 users. CBS_{21} , CBS_{22} and CBS_{24} reach their minimum thresholds and have to be removed and replaced by CBS_{11} , CBS_{12} and CBS_{14} .

3 Formal Modeling of Elasticity of CCBS

The specification of complex systems play an important role in their reliability control. Indeed, they serve as reference for system implementation. The use of formal methods [2] is then the best way to assist the design and validation of these specifications.

More specifically, we use CPN which are an extension of Petri nets(PN). Indeed, PN are used to model the dynamic behavior of discrete systems. They are composed of two types of objects which are: places, that represent the states of the system and contain information represented by tokens, and transitions which represent the events of the system. Places and transitions are related by arcs. However, with PN, it is impossible to model similar behaviors using a single condensed representation. This limitation of PN does not allow us to differentiate between users. That's why we use CPN.

Actually, CPN offer three types of extension which are: extension with time, extension with data and extension with hierarchy. Since we can represent users with tokens, the extension with data allows us to assign information specified by users in their requirements. The extension with time allows to assign a common time to the group of users demanding the CCBS at the same moment. This helps us organize and differentiate between different users.

Below is the formal definition of CCBS using CPN.

Definition 3: $CCBS_i$ is a CPN (P, T, C, E, M_0) where:

- P is the set of places. They represent the states of the CBSs composition.
- T is the set of transitions. Each transition represents a CBS.

Note that: $P \cup T = \phi$ and $P \cap T = \phi$

Each transition has a specific price (the cost of the service). So, we assign to each transition a variable pr_i indicating its price if it is not replicated, else, the price of its replication.

For example, the price of the transition T_{i1} is defined as follows: $\text{val } pr_{i1} = 0.4$;

- C is the set of colours. It defines for every place its colour domain. In our proposal, every place has as type "Info" defined as follows:

- colset Info = product U^*RE^*RE timed; knowing that:
- colset $U = \text{index us with } 0..n$; : every token belonging to $CCBS_i$ has the value $us(i)$. $us(i)$ represents the set of users that can be handled by one $CCBS_i$.
- colset $RE = REAL$; : an integer type. The first RE represents the number of users that can be handled by one CCBS, and the second one represents the sum of their budgets.

Let's note that Info is timed to indicate the evolution of the process through time. For example, the token $(u(1), 25, 26)@1$ indicates that at time 1, we have 25 users whose budget is 26.

- E is an arc expression function. The colour of each arc must be the same colour of the place to which the arc is entering or outgoing.

It is defined as follows: (us,nb,bd) knowing that these variables are declared as follows:

var us: U; var nb,bd: RE; us, nb and bd represent respectively users ID, their number and their budgets.

- M_0 is the initial marking of the net. It describes, in a net, how coloured tokens are situated in different places at a specific time of the execution.

Notation: Let N be a CPN representing $CCBS_i$, $p \in P, t \in T$ and k is the defined arc colour. We note:

- $\bullet t = \{p \in P \mid W^- > 0\}$: Input places of t .
- $t^\bullet = \{p \in P \mid W^+ > 0\}$: Output places of t .
- $Pre : P \times T \rightarrow \{k, 0\}$. If an arc links P_m to T_n , then, $Pre(P_m, T_n) = (us, nb, bd)$, else, $Pre(P_m, T_n) = 0$.
- $Post : P \times T \rightarrow \{k, 0\}$. If an arc links T_n to P_m , then, $Post(P_m, T_n) = (us, nb, bd - pr_{in} * nb)$, else, $Post(P_m, T_n) = 0$.

After having defined the modeling of each $CCBS$, we move now to explain the modeling of elasticity strategies, using the Algorithm 1.

In fact, when a transition T_{ik} of one $CCBS_i$ is not replicated and used from another $CCBS_j$, i.e. $T_{jk} \leftarrow T_{ik}$, then, two cases exist:

- T_{ik} is the first transition of the $CCBS_i$, i.e., $T_{ik} \leftarrow T_{i1}$. In this case, we create a transition named $Tlink_j$ having as input place P_{i0} of $CCBS_i$ and output place P_{j0} of $CCBS_j$. Then, use the non-replicated transition T_{j1} from $CCBS_j$ and link it to the input places of the next transition of $CCBS_i$. This is given by the steps 6–17 in the Algorithm 1.
- T_{ik} is any transition of the $CCBS_i$, except the first one. In this case, we link the previous transition of T_{ik} to the input places of T_{jk} . Then, T_{ik} is linked to the input places of the next transition of T_{ik} , which is $T_{i(k+1)}$. Steps 24 to 36 of the Algorithm 1 model this replication.

Let's note that the replication of only one transition requires the creation of a new $CCBS$.

The deletion action of just one transition implies the deletion of its input places. Its output places will be linked to the same non-replicated transition. However, if all the $CCBS$ will be deleted because of the decrease in number of user demands, then, all its transitions and places will be deleted and the rest of users will be assigned to the previous existing $CCBS$ s. By applying this algorithm, the modeling of the strategies 1 and 2 using CPN Tool is represented by the Fig. 1.

4 Verification of Elasticity Strategies

4.1 Formal Analysis of the Proposed System Modeling

The use of formal methods for software and hardware design is motivated by the welling to achieve the appropriate mathematical analysis, which can contribute to the reliability and robustness of a design. This guarantees safe operation of these critical designs. Indeed, the first question that may arise after designing our model is if this model checks the specification and if it is correct and coherent. It therefore seems essential to check our graph. To do so, we use formal verification.

Indeed, our composition model is a combination of many CCBSSs. Each one of them is a CPN having a specific initial marking. A case in one CCBS starts with a token located in the initial place. After a series of steps, this token evolves towards a final marking that should be located in the final place. So, an important property to be checked in this model is the **reachability** property. In fact, we must ensure that from such an initial marking, it must be possible to reach the final place. This is what we call the **reachability** property of the marking of output places from the marking of input places [11].

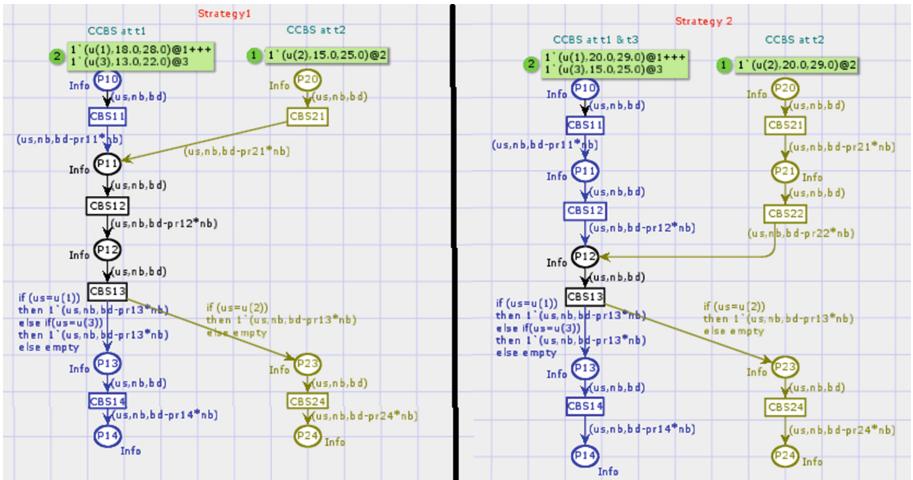


Fig. 1. Modeling of elasticity strategy using CPN tool

Moreover, we should check the absence of dead transitions in every CCBS. That is to say, all transitions can be enabled. This is called the **absence of deadlock** property [12].

In order to express specific properties and verify them, first of all, a generation of state graph must be processed. CPN tool automatically generates and calculates this state graph using a strongly connected components (SCC) graph. Then, we can express properties and query them so that CPN tool checks them. In our case, two properties are checked which are:

Algorithm 1. Modelling of CCBSSs replication using CPN

```

1: for  $CCBS_i$  in  $CCBS_L$  do
2:    $P_{i0} \leftarrow P_{10}$ ;
3:    $P_{im} \leftarrow P_{1m}$ ;
4:    $\sum_{k=1}^p T_{ik} =^* P_{i0}$ 
5:   for j=1 to p do
6:     if ( $T_{ik} \in S_2$ ) then
7:        $Pre(P_{i0}, T_{ik}) = (us, nb, bd)$ ;
8:     else if ( $T_{ik}$  is used from  $CCBS_l$ ) then
9:        $Pre(P_{i0}, T_{link_l}) = (us, nb, bd)$ ;
10:       $Post(P_{k0}, T_{link_l}) = (us, nb, bd)$ ;
11:       $T_{ik} \leftarrow T_{lk}$ ;
12:       $T_{lk}^* \leftarrow (T_{lk}^* +^* T_{i(k+1)})$ ;
13:       $Post(\bullet T_{i(k+1)}, T_{lk}) = (if\ us = u(i)\ then\ 1^*(us, nb, bd-pr_{lk} * nb)$ 
14:        else empty);
15:       $Post(\bullet T_{l(k+1)}, T_{lk}) = (if\ us = u(l)\ then\ 1^*(us, nb, bd-pr_{lk} * nb)$ 
16:        else empty);
17:    end if
18:   end for
19:   for k=(p+1) to n do
20:     if ( $T_{ik} \in S_2$ ) then
21:        $T_{ik} \leftarrow$  Replication of ( $T_{lk}$ );
22:        $\bullet T_{ik} \leftarrow$  Replication of ( $\bullet T_{lk}$ );
23:        $Pre(\bullet T_{ik}, T_{ik}) = (us, nb, bd)$ ;
24:     else if ( $T_{ik}$  is used from  $CCBS_l$ ) then
25:        $T_{i(k-1)}^* \leftarrow \bullet T_{ik}$ ;
26:        $Post(\bullet T_{lk}, T_{i(k-1)}) = (us, nb, bd-pr_{i(k-1)} * nb)$ ;
27:        $T_{ik} \leftarrow T_{lk}$ ;
28:       if ( $T_{ik}^* \neq P_{lm}$ ) then
29:         repeat step 12-16;
30:       else
31:          $T_{lk}^* \leftarrow (T_{lk}^* + P_{im})$ ;
32:          $Post(P_{im}, T_{lk}) = (if\ us = u(i)\ then\ 1^*(us, nb, bd-pr_{lk} * nb)$ 
33:           else empty);
34:          $Post(P_{lm}, T_{lk}) = (if\ us = u(l)\ then\ 1^*(us, nb, bd-pr_{lk} * nb)$ 
35:           else empty);
36:       end if
37:     end if
38:   end for
39: end for

```

- Reachability: This property consists in checking if all output places are reachable. Indeed, the reachability of output places confirms that the process of each CCHS is successfully done. CPN tool offers us a simulation palette to check the execution of different tokens situated in the initial places. Our model was simulated more than 100 times to check if output places are reached. All of these simulations have showed the success reachability of the four output places.

However, this is not enough to confirm that output places are always reachable from the initial marking. So, a query for each output place was executed to confirm this property. The expression of this query is as follows: **SccReachable'(p1,p2)** ; knowing that:

p1: is the initial state of the model, i.e., when input places are marked.

p2: is the final state of a specific CCBS, where the output place of CCBS is marked and all other places do not contain the token of the current CCBS. This query returns either false if the final node is not reachable or true with the specific path if the final node is reachable.

The state of different places is detected from the SCC graph. Let's note that the initial state is 1 and the final states of CCBS1, CCBS2 are respectively: 73 and 87. So, two different queries must be checked which are: **SccReachable'(1,73)** and **SccReachable'(1,87)**. Both of these two queries were successfully checked and the result is shown in Fig. 2.

- Absence of deadlock: A deadlock corresponds to a CPN marking in which no more transition is allowed. So, there must be no dead transitions. This can be checked in CPN tool using the following query: **ListDeadTIs()**. When executing this query, we can have two possible results: even a list of dead transitions or an empty list. In our model, this query returned an empty list, as shown in Fig. 2, a result that confirms the non-existence of deadlock.

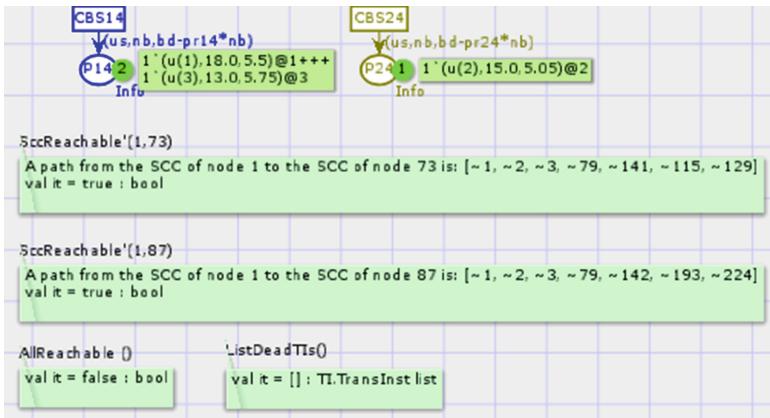


Fig. 2. Formal verification of CCBSs using CPN tool

Thus, we can confirm that the proposed modelling is valid. It does not contain any deadlock and reaches always the final states.

5 Related Work

Cloud Elasticity is a highly studied topic. Several mechanisms have been proposed to ensure it. However, we focus on works which were proposed to minimize the cost when applying elasticity.

Liu et al. [9] proposed an algorithm allowing them to minimize the cost of used SaaS by decreasing the unused Virtual machines from the IaaS. This algorithm aims at not violating the performance provided to the final user.

Wu et al. [15] proposed a system that maximizes the accepted number of users requesting a certain SaaS. This is done by an efficient placement of requests on Virtual machines offered by different IaaS providers. In fact, they proposed an algorithm that maximises the use of already initiated VMs so that many users can access them after being classified according to the waiting time. This solution is a cost benefit for the provider perspective.

Han et al. [5] have also focused on cost reduction when applying an elastic scaling approach of multi-tier applications in Cloud computing. Indeed, they proposed an approach that detects the bottlenecks in a class of these applications so that they can accordingly scale up and down resources at these points.

The above cited works are dealing with the vertical elasticity whereas we are handling horizontal one. Besides, the proposed models were not checked, which is a very important task to ensure their validity.

However, Narkos et al. [10] proposed a model, based on Markov Decision Chain, allowing the automatic elasticity by increasing and decreasing the number of Virtual machines. Indeed, the action that should be processed to ensure the elasticity operation is checked and expressed using PCTL. Besides, the reachability property is checked in this model using PRISM tool. This work treats also the vertical elasticity, which is not the case for our work. Moreover, we are handling the composite Cloud services and not the atomic Cloud services.

It is in this sens that the work of Amziani et al. [1] is oriented. Indeed, they proposed a controller to check the behavior of service-based business process in the Cloud when applying elasticity operations. The controller is modeled by high level Petri nets. Time and maximum and minimum thresholds of one service are the main indicators of elasticity actions. Their work was a start point for us but our work differs from them in three main points which are:

- Additionally to the threshold, our approach focuses on cost when applying elasticity actions.
- Our approach checks the validity of the obtained model before comparing between strategies.
- The modeling that we propose allows even the modeling of just one Cloud service in case it is required atomically.

6 Conclusion

Cloud computing has proven to be more secure, more reliable, more scalable and more affordable than traditional IT. These are some of the Cloud characteristics that made it widely used in many fields. As a result, many sectors of this field used the Cloud architecture to offer their services that must be composed to perfectly meet the demand of their users, whose number is more and more increasing. So, a replication of the composite service, called CCBS, must be processed to answer to user queries at the same time. Some strategies were

proposed in this context. Our contribution in this paper was to check the validity of these strategies using formal models and to compare between them. To do so, our composition was modeled using CPN and was validated by checking the reachability and the absence of deadlock properties.

However, CPN tool does not allow us to check specific properties. So, as a future work, we propose to check soundness and temporel properties using a suitable tool. Besides, we intend to test this mechanism with real CBSs and to implement a tool allowing the automatic execution of elasticity actions.

References

1. Amziani, M., Melliti, T., Tata, S.: Formal modeling and evaluation of service-based business process elasticity in the cloud. In: 22nd IEEE International Conference on Collaboration Technologies and Infrastructure (WETICE 2013), pp. 284–291. Hammamet, Tunisia, June 2013
2. André, P.: Méthodes formelles et à objets pour le développement du logiciel: études et propositions (1995)
3. Guerfel, R., Sbaï, Z., Ayed, R.B.: On service composition in cloud computing: a survey and an ongoing architecture. In: IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014, Singapore, 15–18 December 2014, pp. 875–880 (2014)
4. Guerfel, R., Sbaï, Z., Ayed, R.B.: Towards a system for cloud service discovery and composition based on ontology. In: Núñez, M., Nguyen, N.T., Camacho, D., Trawiński, B. (eds.) ICCCI 2015. LNCS (LNAI), Part II, vol. 9330, pp. 34–43. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24306-1_4
5. Han, R., Ghanem, M.M., Guo, L., Guo, Y., Osmond, M.: Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Future Gener. Comput. Syst. **32**, 82–98 (2014)
6. Herbst, N.R., Kounev, S., Reussner, R.H.: Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), pp. 23–27. USENIX, San Jose (2013)
7. Islam, S., Lee, K., Fekete, A., Liu, A.: How a consumer can measure elasticity for cloud platforms. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE 2012, pp. 85–96. ACM, New York (2012)
8. Jensen, K.: Coloured petri nets: a high level language for system design and analysis. In: Rozenberg, G. (ed.) ICATPN 1989. LNCS, vol. 483, pp. 342–416. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-53863-1_31
9. Liu, Z., Wang, S., Sun, Q., Zou, H., Yang, F.: Cost-aware cloud service request scheduling for saas providers. Comput. J. **57**(2), 291–301 (2014)
10. Naskos, A., Stachtiari, E., Gounaris, A., Katsaros, P., Tsoumakos, D., Konstantinou, I., Sioutas, S.: Cloud elasticity using probabilistic model checking. CoRR, abs/1405.4699 (2014)
11. Sbaï, Z., Barkaoui, K., Boucheneb, H.: Compatibility analysis of time open workflow nets. In: International Workshop on Petri Nets and Software Engineering (PNSE 2014), CEUR Workshop Proceedings, vol. 1160, pp. 249–268, June 2014. <http://ceur-ws.org/Vol-1160/>
12. Sbaï, Z., Guerfel, R.: CTL model checking of web services composition based on open workflow nets modeling. IJSSMET **7**(1), 27–42 (2016)

13. Inc Sun Microsystems.: Introduction to cloud computing architecture. Technical report, June 2009
14. CPN tool. <http://cpn-tools.org/>
15. Wu, L., Garg, S.K., Buyya, R.: SLA-based admission control for a software-as-a-service provider in cloud computing environments. *J. Comput. Syst. Sci.* **78**(5), 1280–1299 (2012)



QoS-Driven Self-adaptation for Critical IoT-Based Systems

Arthur Gatouillat¹✉, Youakim Badr¹, and Bertrand Massot²

¹ Univ Lyon, INSA Lyon, LIRIS, UMR5205, Lyon, France

{arthur.gatouillat,youakim.badr}@insa-lyon.fr

² Univ Lyon, INSA Lyon, INL, UMR5270, Lyon, France

bertrand.massot@insa-lyon.fr

Abstract. The Internet-of-Things, which designates the interconnection of numerous physical devices, is a growing research direction faced with many challenges. One of these challenges is to provide constant quality-of-service despite IoT devices being used in a constantly changing physical environment. In order to answer this problem, we introduce a quality-of-service driven self-adaptation framework, which can simultaneously handle changing adaptation strategies, monitoring infrastructure and physical environment while guaranteeing constant quality-of-service. Because of its formal guarantees, our system is particularly suited for the control of critical IoT-based systems, and we thus demonstrated its practicality by applying it to an e-health case-study where the safety of the monitored patients must be assured.

Keywords: Self-adaptive systems · Adaptive IoT · Controller synthesis

1 Introduction

The Internet-of-Things (IoT) enables the interconnection of virtually every object of the physical world, such as a variety of sensors, actuators, robots or wearable devices. This interconnection of various physical-world devices into IoT systems, coupled with the strong hardware constraints of IoT devices, mandates the study of adaptation strategies to deal with devices failure, especially when dealing with critical systems (e.g. healthcare systems). To deal with such requirements, self-adaptation software and autonomic frameworks were proposed [1–4]. In particular, self-adaptive software systems (SAS), which deal with distributed applications in changing environments, normally require human supervision to sustain despite changes during their executions. These systems rely on a closed feedback loop to adjust themselves to changes. Based on observed context variables and thresholds, they monitor themselves, their context and entities from the target system environment to decide when and how to apply adaptation strategies in order to ensure expected behavior (i.e., functional requirements) and guarantee quality of services (i.e., non-functional requirements) [5].

In the context of the Internet-of-Things, self-adaptation is a salient property of smart objects. It allows them to be self-configured and adapted to extreme conditions while ensuring the target system objectives such as comfort, automation, security and safety goals. Self-adaptation mechanisms driven by adaptation goals modify smart

objects behavior dynamically. However, the IoT is a dynamic and global network infrastructure, in which “things” are expected to be autonomous and self-configurable. This feature that the IoT should strive to achieve is not a trivial task since adaptation goals may also evolve continuously during run-time due to changes in functional or non-functional requirements and contextual information. These changes affect observed variables, their thresholds and even alter the monitoring logic when context variables are added or deleted. As a result, when adaptation goals change at run-time, both monitoring and self-adaptation mechanisms may become inapplicable because either the adaptation mechanism deals with outdated goals, or the monitoring mechanism addresses monitoring requirements that are irrelevant to the actual adaptation goals.

In this paper, we present a discrete controller synthesis system that ensures self-adaptation behavior based on quality-of-service properties (QoS) of smart objects during their run-time. Our system is resource-aware and ensures simultaneously the separation of concerns of adaption objectives, context monitoring and adaptation strategies. By such, our system makes possible to guarantee the service level agreement (SLA) which defines the level of smart object services as expected by end-users expressed them as constrains on non-functional properties. Without loss of generality, our system focuses on the safety quality as a crucial non-functional property in the context of healthcare monitoring to detect heart malfunctions with wearable sensors (heart rate sensor and electrodermal activity sensor) and ambient sensors (occupancy detection sensor, noise sensor, etc.). In this context, we model the safety quality as a qualitative property in the SLA of wearable and ambient sensors. The safety quality is initialed defined as the resource-awareness factor of wearable sensors in response to their battery consumption levels. During the execution, the safety quality may be renegotiated by adding resilience as an additional new factor. This new contracted factor states that the defected objects are subsumed by alternative smart objects (same types) or inferring their observed values through data collected from other smart objects (different types). Yet another adaptation of the safety property may include the health-awareness factor, which enables the detection of critical health crisis from contextual information, medical sensor values, and patient history patterns. Our discrete controller synthesis system implements a dynamic monitoring approach that deploys, at run-time, new context gatherers and new monitoring requirements for new quality properties. These elements are automatically generated from the non-functional quality-of-service properties in the SLA and deployed at run-time without interrupting the target system execution or the adaptation mechanism.

The remaining of the paper is organized as follows. In Sect. 2, we present related works. Before introducing our discrete controller synthesis system, we briefly introduce a motivation case-study dealing with safety quality in wearable sensors in Sect. 3. In Sect. 4, we present our QoS-driven self-adaptation approach which is based on the DYNAMICO reference model [3] and includes SLA and stated-based failure ontologies. These ontologies along with knowledge-based rules are used to generate non-functional device labeled transition graphs and to generate our discrete controller synthesis system as described in Sect. 5. In Sect. 6, we present our implementations and conclude our work in Sect. 7.

2 Related Works

The work detailed in this paper is located at the intersection of three distinct but related communities: home automation, classical control theory and software auto-adaptation.

The home automation community deals with the integration of smart devices such as sensors, actuators and gateways into houses in order to better monitor and control living environments. Home automation is a broad concept, including smart home and ambient intelligence (AmI), which generally refers to architectures, practices, and controllers for proper management of the home life-cycle to address home safety, energy efficiency, entertainment, ambiance, assisted living, fall detection, elderly care or patient monitoring [6]. Controllers are key components of home automation. Rule-based controllers have been widely explored in an AmI context and more particularly in the context of remote health-monitoring; fuzzy rules were mixed with case reasoning in [7] to provide both home-automation and health monitoring to elderly patients. Ontology-derived rules are also used in coordination with rule engines to provide functional intelligent building behavior [6]. Rule-based framework in the context of remote monitoring where explored by [8] to provided assistance in decision-making for healthcare providers. Many contributions in the field of rule-based home automation have focused on the management of functional properties, but they lack consideration of system non-functional properties and adaption to situation where some sensors are failing but the system must still perform. The main concern of these contributions is the functional coordination of distributed sensors and actuators, instrumenting a house in order to achieve predefined goals. A rule language approach allows, thanks to its relative expressiveness, the specification of control objectives that can then be executed using rules engines.

The software adaptation community deals with the integration of strategies enabling better handling of changing digital and physical environment to modular software systems. These contributions can be divided into three categories: contributions which consider the adaptation of classical control frameworks to software systems, contributions which consider the use of monitor analyzer planner executor and knowledge feedback (MAPE-K) loops [9] as the basic building block to enable software adaptation, and finally contributions combine classical control and MAPE-K loops to implement adaptive software solutions.

When only classical control solutions are applied to adaptation problems, the main research challenge resides in the accurate state space modelization of software processes. MPC-based [10] and PID-based [11] adaptation framework were able to provide results in terms of software adaptation. However, we believe that the cumbersome modeling process that must occur for each software system is not suitable for IoT applications. Indeed, self-adaptation for the IoT must be able to handle potentially very heterogeneous devices, that are not easily modeled using state space representations. Indeed, the feedback loop modelization of classical control systems does not provide good modularity, as elements of the feedback loop are not standard, and can vary between systems. Standardization of the feedback loop elements is provided by the MAPE-K self-adaptation framework.

MAPE-K is considered as a gold-standard for self-adaptive systems [3, 4, 12]. The idea behind the MAPE-K control scheme is to define autonomic elements defining an adaptation loop from monitors (i.e. sensors) to executors (i.e. actuators) that perform system reconfiguration using knowledge shared between all the feedback loop elements. Because this feedback loop is only a reference model, multiple implementations have been studied such as agent-based implementations [12] or using formal frameworks such as FORMS [4]. However, such control feedback loops are purely software based, and are not appropriate for the control of hybrid software-hardware systems such as IoT systems. Contributions considered the mixed use of classical control loops and MAPE-K control loops to enable software systems with self-adaptation properties. This is the case of DYNAMICO [3], an architectural reference model equipped to deal with changing system requirements and monitoring infrastructure thanks to separation of concerns. In this architecture, three independent MAPE-K loops are interconnected using classical control feedback loops in order to deal with changing system objectives, changing monitoring framework and system adaptation. This reference model is however oriented at purely software self-adaptive systems, and mandates some modifications to be used in the context of hybrid IoT systems. Another field of interest when it comes to control theory is discrete controller synthesis (DCS). In this field, discrete models of target to-be-controlled systems are used to build correct-by-construction discrete controllers [13]. Such control strategy has been successfully used in the IoT context for functional adaptation of simple home-automation systems [1]. The control objectives in this contribution are given as first order-logic rules, and some non-functional concerns are integrated under the form of controlled energy consumption. The strategy behind such control framework is to divide the systems into a set of controllable and non-controllable states, and to use the controllable states to guarantee system objectives fulfilment given non-controllable states. Originally, such discrete control systems were built using synchronous programming languages such as SIGNAL [13] or BZR [1, 14, 15]. Event-condition-action rules based discrete controller synthesis was explored, more particularly using an event-condition-action (ECA) rules based high level description language to BZR translation to perform controller synthesis [16, 17]. Asynchronous controller synthesis methods in the context of cloud-based autonomic manager was studied using the LNT framework [18]. The main advantage about these contributions is that the controller verification part can be avoided because controller synthesis is assumed to produce a correct controller. Table 1 summarizes all the contributions described in this section.

Table 1. Related contribution synthesis.

Community	Concerns	Tools	Papers
Self-adaptive software	Enabling software to feature adaptive behavior to changing environment	MAPE-K, DYNAMICO, FORMS	[3, 4, 9, 12]
Discrete controller synthesis	Build correct controllers using discrete models of controlled systems	BZR/Heptagon, LNT, Signal	[1, 14–18]

Our contribution is at the center of all the before-mentioned contributions since it adapts the DYNAMICO reference model to the context of IoT and uses MAPE-K and classical feedback control loops to enable the capability of dealing with changing system objectives and monitoring infrastructures. By using a state-chart model of non-functional properties and high-level ECA-rules, we generate IoT system controllers with well-established DCS tools such as the Heptagon/BZR toolbox.

3 Motivation Case-Study

We consider the surveillance of a patient at risk of myocardial infarction recurrence as a motivation case study. In this context, the patient requires continuous monitoring of physiological parameters to detect potential recurrences and to urge a rapid medical response if a heart failure is detected. Continuous monitoring is achieved using wearable wireless sensors, which are battery powered, resulting in different resources constraints. In addition, the living environment of the patient is instrumented with sensors and actuators that are either continuously powered or battery powered. As the case of most IoT-based applications, connected objects are strongly constrained in terms of resources: all sensing and actuation devices feature limited computing abilities (CPU frequency up to a few hundreds of megahertz), storage (up to a few megabytes) and volatile memory (up to a few hundreds of kilobytes). Constrained resources also imply limitations in terms of communication protocol, which must be lightweight in order to avoid the introduction of processing overhead and limit energy consumption.

In this case-study, we particularly focus on the robust detection of heart malfunctions, and the triggering of emergency medical response if such a situation occurs. There are two main robustness requirements: the avoidance of false positive detection of heart malfunction (i.e., the system detects a cardiac malfunction while there is none), but more importantly the detection of cardiac failure even if the system does not operate at full capacity.

Considering self-adaptive properties of such a system, this case-study is of particular interest: the adaptation goal is to ensure the safety property while satisfying quality of service of a continuous and reliable monitoring. To satisfy the safety goal, the adaptation strategy is based on the resource-awareness factor, resilience factor (i.e., substitution of defected objects with alternatives) and healthcare awareness factor such the request for medical assistance (i.e., myocardial infarction detection) or technical intervention (i.e., abnormal values). Consequently, a safety-enabled smart home is implemented to support self-adaptation objectives based on resources consumption, resilience and external assistance. The adaptation strategies (i.e., the mechanisms that affect the target system) consist of modifying smart sensor parameters based on resource monitoring, substituting defected objects with alternatives or inferring their values from nearby smart-objects as well as call for medical assistance when detecting abnormal values. The context variables mandating observation are battery levels, absence or abnormal values, exceeding medical thresholds that define myocardial infarction.

In the following sections, we limit ourselves to examples using only three sensors: the battery-operated heart rate (HR) and heart rate variability (HRV) sensor,

the battery-operated electrodermal activity (EDA) sensor, and the continuously powered occupancy sensor. The cardiac risk is thus evaluated by a cardiac health estimation service including inputs:

- Continuous streams of HR and HRV values (optimal mode)
- Continuous streams of EDA values with instantaneous or average HR values (failsoft mode)
- Continuous streams of presence values (critical mode)

Depending on these inputs, the monitoring controller uses its internal cardiac health estimator model to infer cardiac health status and request medical help for cardiac malfunctions. Different cardiac health estimation models are out of scope since we only focus on how the controller is self-adapted to provide estimators with correct inputs at all time, and how it triggers external tiers notification, if deemed necessary.

4 QoS-Driven Self Adaptation

4.1 Managing Changing SLA and Monitoring Environment

Purely functional and static adaptation was successfully studied in IoT [1, 17]. However, insuring that IoT-based systems behaves as specified under changing control objectives of non-functional properties and limited resource-awareness is a relatively unexplored research field. The DYNAMICO reference model [3] provides a prominent solution to design and implement self-adaptive software systems where both adaptation and monitoring infrastructures are enabled with self-adaptive capabilities using three types of feedback-loops:

- The *objectives feedback loop*, which governs changes in adaptation goals, also called control objectives (e.g. SLAs);
- The *target system adaptation feedback loop*, which regulates the target system requirements satisfaction and the preservation of the adaptation properties;
- The *dynamic monitoring feedback loop*, which infers context variables from the contracted quality of service (QoS) conditions and adapts the architectural reconfiguration of the monitoring infrastructure to implement the monitoring logic associated with context variables.

DYNAMICO characterizes the separation of concerns and interactions among different types of feedback loops. Despite its prominent advantages, DYNAMICO implementations (i.e., SMARTERCONTEXT monitoring infrastructure with the QoS-CARE/FRASCATI middleware) are not relevant to distributed smart-objects with limited resources.

Based on the based on the DYNAMICO reference model, we propose an IoT-targeted self-adaptive system, including a distributed adaptation infrastructure and the controlled IoT system, consisting of gateways, each of which interacts with one or more devices or sensors (see Fig. 1). Gateways invoke device services to get their data streams, adjust their functional parameters, monitor their non-functional properties and perform adaptation operations when deemed necessary, as defined in their SLAs.

More precisely, the IoT-based self-adaptive system relies on a SLA for each sensor, and a global SLA for the system as a whole to respectively set adaptation objectives and deploy monitors to gateways to observe each sensor's QoS. Monitors informs the discrete controller with events to decide whether that a self-adaption strategy should be applied to meeting end-users' SLAs. The adaptation infrastructure in Fig. 1 illustrates the causal relationships between adaptation objectives, monitors and discrete controller, and interactions between different feedback loops.

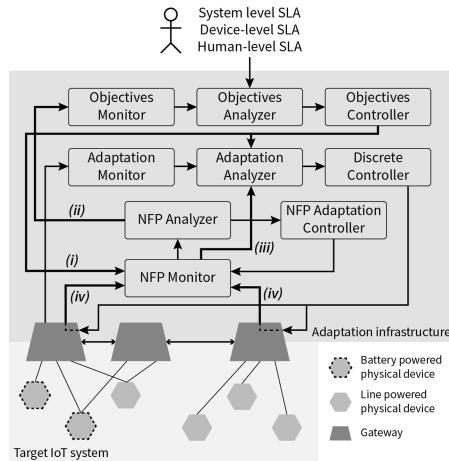


Fig. 1. An IoT-based self-adaptive system based on the DYNAMICO reference model

Interactions (i) between the objectives feedback-loop and the adaptation feedback-loop (the monitoring feedback-loop resp.): These interactions feed the reference control input computed by the objectives controller to the adaptation loop and the monitoring loop. For instance, in our use-case, the first resource-aware adaptation to be considered is related to the sensors' battery levels by which the reference input will thus be under the form $\text{BatteryLevel} > 20\%$. This reference will be used by the adaptation feedback loop as an element to be analyzed to decide potential adaptation, and by the monitoring feedback loop as a reference input for context monitoring.

Interactions (ii) between the Monitoring Feedback Loop and the Objectives Feedback Loop: These interactions characterize the detection of the need of a change in the control objectives by the monitoring feedback loop to be fed to the control objectives feedback loop. In our use case, if the battery is drained, and if the cardiac status monitoring must be inferred using other environmental sensors. This mandates a change of control objectives that must be decided by the objectives feedback-loop.

Interactions (iii) between the Monitoring Feedback Loop and the Adaptation Feedback Loop: These interactions feeds adaptation-triggering events from the monitoring loop to the adaptation loop. For instance, the consistent and more rapid than normal decrease of the battery level can be used as an event to trigger faster adaptation

of the system and a quicker adoption of a battery-saving fail-soft mode in order to extend the duration of quasi-optimal system behavior.

Interactions (iv) between the Adaptation Feedback Loop and the Monitoring Feedback Loop: These interactions feed the internal context from the adaptation loop to the monitoring loop. This interaction can be used to insure system consistency after adaptation. For instance, in our use case and if the HR sensor has to be subsided by position sensor for health monitoring, this interaction is used to ensure that the position sensors are all in a functional state after the adaptation, thus insuring global system safety.

Figure 2 describes the self-adaptation meta rules of the target IoT system behavior in response to changes in the SLAs (i.e. control objectives' changes) through interaction (*i*) and to changes in the monitoring infrastructure (e.g. sensor removed from network) through interaction (*ii*).

```

If new control objective (SLA renegotiated)
  Perform objectives analysis
  Generate new control contract
  Deploy monitors in gateways
  Perform discrete controller synthesis
If change in context (devices not available)
  Generate new control contract
  Deploy monitors in gateways
  Perform discrete controller synthesis

```

Fig. 2. Self-adaptation meta rules

4.2 Non-functional Device Labeled Transition Systems

As illustrated in Fig. 3(a), we propose an ontology to describe the global SLA for the IoT target system. Each non-functional property (i.e., safety) is thus defined in terms of QoS factors (resource-awareness, resilience, healthcare awareness) each of which has constraints expressed as service level objectives (SLO) and has a corresponding monitor deployed on the gateway of the sensors related to each QoS factor. We also propose a failure-adaptation ontology to describe the system global safety in terms of resources (i.e., low battery), resilience (unattached sensor, interrupted communication), abnormal data due to critical heart failures or hardware failures (digital/analog failures).

A building block of our IoT-based self-adaptive system is the description of non-functional behaviors of each sensor using labeled state transition systems (LTS) based on the failure-adaption strategies' ontology. In fact, each state corresponds to a failure and transitions refer to adaptation strategies to ensure safety non-functional property of the target IoT system. This description allows the expression a qualitative quality-of-service in terms its quantitative factors which are observed with monitors at run-time and makes a correlate with appropriate adaptation strategies in case of failures. It also provides a discrete behavior using states and transitions, making possible to use toolbox (i.e., Heptagon/BZR) for synthesizing discrete controllers.

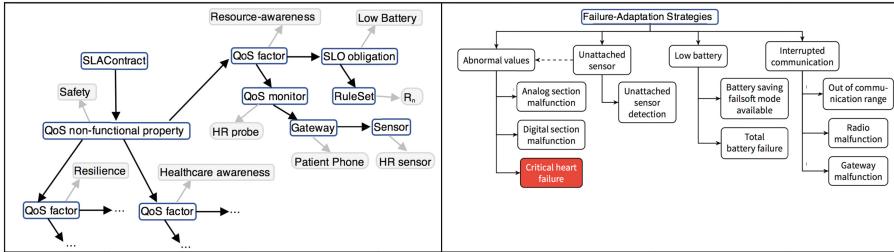


Fig. 3. (a) Global QoS ontology (b) Failure-adaption strategies ontology

Generally speaking, a LTS is defined as a quadruple $(S, L, \rightarrow, s_{in})$, where S is a set of states, L a set of transition labels, $\rightarrow \subseteq S \times L \times S$ is a transition relation between two states, and s_{in} is an initial state. The set of transition labels is defined as $L = (events, actions, \setminus)$, where $\setminus \subseteq events \times actions$.

LTSs enable an accurate system description, where non-functional and non-controllable variables are members of the set $events$, and services calls are included in the set $actions$. This syntactic separation of non-functional and functional variables enables better expressivity: the statement “ $e \setminus a$ ” can be interpreted as the control of event e by service a when event e during the firing of the transition. When no control service is provided for a given transition, it implies the non-controllability of the given transition. Battery operated ‘smart’ sensors are purely uncontrollable (because the system cannot automatically charge the battery, external human intervention is required for this operation). Figure 4 describes the non-functional behavior of the heart-rate sensor used in our case study. It features both purely event-based and event\action transitions. It is worth noticing that while some non-functional states can be self-detected by the sensor itself (such as the low-battery state or the unattached state), some other can only be inferred by the controller on a more global view (e.g. the analog malfunction state, which is accessed through the abnormalValue transition event). Because of space limitation, we will not detail what is considered as an abnormal value, nor will we detail the LTS of other sensors, which are assumed to be similar to the heart-rate sensor. The choice of LTS as a model of computation (MoC) of IoT-based devices was motivated by the genericity of this MoC, and because IoT-based devices traditionally present a discrete state-based behavior (e.g. a sensor can be on, measuring, off, etc.). Consequently, both personal and external-tier devices can be represented using this MoC.

4.3 Rule Based Modelization of Control Objectives and SLA

Since our self-adaptive approach is based on the DYNAMICO reference model, we express the control objectives in terms of rules. Rule-based definitions of objectives is declarative and easily express desired control objectives by end-users. The discrete controller (see Fig. 1) will be provided with rule-based control objectives to decide whether non-functional properties are guaranteed. Please note that SLA non-functional properties are considered to be control objectives and they are provided as inputs to the

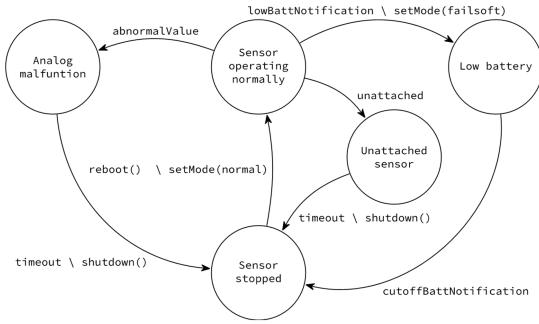


Fig. 4. Non-functional LTS of the Heart Rate sensor

self-adaptive system. A rule refers to a condition-assertion statement and is defined as follows:

IF condition ASSERT action

Figure 5 briefly illustrates couple of rules for safety non-functional property. The goal of these rules is to make sure that if either of the cardiac sensor is not operating normally or the EDA sensor is shut down, the position sensor is turned on to allow a better health status estimation.

```
R1: IF Not(HR.state == normal) ASSERT Pos.state == normal
R2: IF EDA.state == shutdown ASSERT Pos.state == normal
...
Rn: IF HR.Battery == Low ASSERT HR.setMode(FailSoft)
```

Fig. 5. Control objectives for a resilient cardiac monitoring

5 Synthesizing the Discrete Controller

The labeled transition system description of each sensor in terms of functional and nonfunctional properties leads to a set of distributed systems that run concurrently. In this context, these descriptions are equivalent to BZR control contracts, that are composed of three elements: an *assumption* (keyword `assume`), an *enforcement* (keyword `enforce`) and a declaration of controllable variables (keyword `with`). Reader may refer to [14] for a complete description the Heptagon/BZR language. In our work, we propose to generate the discrete controller by firstly mapping label transition descriptions of all sensors into a Heptagon/BZR programs and secondly synthesizing them into a discrete controller. Indeed, input of sensor services can be seen as controllable variables where rules can easily be converted into first-order logic enforcements. Because Heptagon is a synchronous programming language, the synchrony of our target IoT-system must be clearly defined. In our context, the synchrony hypothesis states:

given a system, a set of inputs and a set of outputs, the system must be able to compute all of its outputs between two occurrences of inputs changes events [19].

Considering our use-case where computational abilities of gateways (used as centralized controllers) are much greater than computational resources of sensors, the IoT target system tends to behave as a synchronous system. Indeed, because of their gateway processing speeds, controllers running on gateways will be able to process all sensor events before receiving new events. Please note that while this kind of behavior is true for smaller IoT systems such as our wearable and ambient system (made of tens of devices), this assumption might not hold for much bigger IoT-based systems made of thousands of devices, considering the mass of generated events is much higher.

6 Implementation and Simulation Results

As a preliminary simulation, the LTS of the cardiac parameters sensor introduced in Fig. 4 was encoded into BZR along with the LTS of both the position and EDA sensors. A rule set, including rules introduced in Fig. 5, were translated into BZR contracts, and were then used to successfully synthesize a controller using SIGALI. The simulation was performed using the simulator included with the Heptagon/BZR compiler, which demonstrates a correct behavior of the controlled IoT-system with respect to the provided control objectives. Figure 6 is a chronogram presenting the behavior of our controlled system in terms of normal states. For example, we can easily see that both rule r_1 and r_2 are respected. In fact, when the cardiac sensor leaves the normal mode (i.e. it goes into any of other failure or shutdown states), the position sensor is turned on. Similarly, when the EDA sensor leaves the normal state (i.e. it is turned off, since this sensor is modeled as a binary-state sensor), the position sensor stays on.

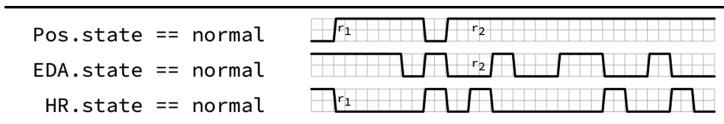


Fig. 6. Excerpt of the simulation chronograms

7 Conclusion and Perspectives

In this paper, we introduce a QoS-driven approach to self-adaptive IoT systems based on the DYNAMICO adaptation reference model and non-functional properties. Our IoT-based system relies on a set of rules and labeled state transitions to generate a discrete controller synthesis. Current ongoing work focuses on better sensor modeling by decoupling functional and non-functional behavior into distinct but interacting LTS to have better separation of concerns (non-functional LTS typically appearing in the monitoring feedback loop, while the functional LTS represents system functional adaptation). Integration of a domain specific rule-based language is also under

investigation. This will enable us to express control objectives and thus improve separation of concerns between objectives feedback loop and the adaptation feedback loop.

Acknowledgement. This work is generously supported by Auvergne-Rhône-Alpes Region research grant.

References

1. Zhao, M., Privat, G., Rutten, É., Alla, H.: Discrete control for the internet of things and smart environments. In: 8th International Workshop on Feedback Computing, San Jose, CA, USA, 25 June 2013
2. Zhao, M., Privat, G., Rutten, E., Alla, H.: Discrete control for smart environments through a generic finite-state-models-based infrastructure. In: Aarts, E., et al. (eds.) AmI 2014. LNCS, vol. 8850, pp. 174–190. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-14112-1_15
3. Villegas, N.M., Tamura, G., Müller, H.A., Duchien, L., Casallas, R.: DYNAMICO: a reference model for governing control objectives and context relevance in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A. (eds.) Software Engineering for Self-Adaptive Systems II. LNCS, vol. 7475, pp. 265–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_11
4. Weyns, D., Malek, S., Andersson, J.: FORMS: a formal reference model for self-adaptation. In: Proceedings of the 7th International Conference on Autonomic Computing, pp. 205–214 (2010)
5. Villegas, N.M., Müller, H.A., Tamura, G., Duchien, L., Casallas, R.: A framework for evaluating quality-driven self-adaptive software systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 80–89 (2011)
6. Bonino, D., Corno, F.: Rule-based intelligence for domotic environments. Autom. Constr. **19**, 183–196 (2010)
7. Yuan, B., Herbert, J.: Context-aware hybrid reasoning framework for pervasive healthcare. Pers. Ubiquit. Comput. **18**, 865–881 (2014)
8. Augusto, J.C., McCullagh, P., McClelland, V., Walkden, J.A.: Enhanced healthcare provision through assisted decision-making in a smart home environment. In: 2nd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (2007)
9. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**, 41–50 (2003)
10. Angelopoulos, K., Papadopoulos, A.V., Silva Souza, V.E., Mylopoulos, J.: Model predictive control for software systems with CobRA. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 35–46 (2016)
11. Peng, X., Chen, B., Yu, Y., Zhao, W.: Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop. J. Syst. Softw. **85**, 2707–2719 (2012)
12. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 13–23 (2015)
13. Marchand, H., Bournai, P., Borgne, M.L., Guernic, P.L.: Synthesis of discrete-event controllers based on the signal environment. Discret. Event Dyn. Syst. **10**, 325–346 (2000)

14. Delaval, G., Marchand, H., Rutten, E.: Contracts for modular discrete controller synthesis. *ACM Sigplan Not.* **45**, 57–66 (2010)
15. Delaval, G., Rutten, E., Marchand, H.: Integrating discrete controller synthesis into a reactive programming language compiler. *Discret. Event Dyn. Syst.* **23**, 385–418 (2013)
16. Cano, J., Delaval, G., Rutten, E.: Coordination of ECA rules by verification and control. In: Kühn, E., Pugliese, R. (eds.) COORDINATION 2014. LNCS, vol. 8459, pp. 33–48. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43376-8_3
17. Cano, J., Rutten, E., Delaval, G., Benazzouz, Y., Gurgen, L.: ECA rules for IoT environment: a case study in safe design. In: Proceedings of the 8th International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 116–121 (2014)
18. Abid, R., Salaün, G., De Palma, N.: Asynchronous synthesis techniques for coordinating autonomic managers in the cloud. *Sci. Comput. Program.* **146**, 87–103 (2017)
19. Gamatié, A.: Synchronous programming: overview. In: Designing Embedded Systems with the SIGNAL Programming Language, pp. 21–39. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-0941-1_2

IoT Systems Provisioning and Management for Context-Aware Smart Cities

Introduction to the 2nd Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities ISYCC'17

The ISYCC'17 workshop was held in conjunction with the 15th International Conference on Service Oriented Computing (ICSOC 2017) on November 13–16, 2017 in Malaga, Spain. This workshop offered an exciting and highly interactive opportunity to show research prototypes in the Internet of Things, (Mobile) Cloud Computing, Distributed and Cloud-Aware IoT systems, Context-Awareness and (Mobile) Crowd-sensing. It was also interesting to see how researchers are applying these paradigms in real environments and situations within a smart city, such as smart transportation, pollution, smart tourism, well-being. We received 15 submissions, out of which 8 were accepted. This workshop clearly showed interesting research efforts in the field and offered the ability to fruitful discussions:

- BiAgent-based Model for IoT Applications : Case of a Collision Avoidance System
- Seamless interactions on the Internet of Things. A Spotify-based proof of concept
- A Feedback-based Adaptive Service-Oriented Paradigm for the Internet of Things
- QoS Prediction for Reliable Service Composition in IoT
- Checking and Enforcing Security through Opacity in Healthcare Applications
- Power-Based Device Recognition for Occupancy Detection
- Cognitive Determination of Policies for Data Management in IoT Systems
- A Research Perspective on Fog Computing

We would like to thank the authors for their submissions, the program committee for their reviewing work, and the organizers of the ICSOC 2017 conference for their support which made this workshop possible.

Organization

Workshop Program Chairs

Javier Berrocal

Luca Foschini

Mohamed Mohamed

Sami Yangu

University of Extremadura, Spain

University of Bologna, Italy

IBM Research, USA

Concordia University, Montreal, Canada

Workshop Committee

Mohamed Abu-Lebdeh

Sabeur Aridhi

Nejib Belhadj-Alouane

Sami Bhiri

Juan Boubeta-Puig

Carlos Roberto De Rolt

Ahmad Dhaini

Walid Gaaloul

Jaime Galán-Jiménez

José García-Alonso

Michele Girolami

Tarek Hamrouni

Mohamad I Jaber

Nikko Mäkitalo

Carla Mouradian

Stefan Nastic

Romain Rouvoy

Peter Ruppel

Ahmed Samet

Samir Tata

Victoria Torres

Mauro Tortonesi

Asma Trabelsi

Zhangbing Zhou

Javier Cubo

Andrea Delgado

Alfonso Garcia-de-Prado

Laura González

Sam Guinea

Kai Jander

Concordia University, Canada

University of Lorraine, France

University Tunis ElManar, Tunisia

University of Monsatir, Tunisia

University of Cádiz, Spain

Universidade do Estado de Santa Catarina, Brazil

American University of Beirut, Lebanon

Telecom SudParis, France

University of Extremadura, Spain

University of Extremadura, Spain

Italian National Research Council, Italy

University of Manouba, Tunisia

American University of Beirut, Lebanon

University of Helsinki, Finland

Concordia University, Canada

TU Wien, Austria

University of Lille, France

Technische Universität Berlin, Germany

Oxford Brookes University, UK

IBM Research, USA

Polytechnic University of Valencia, Spain

University of Ferrara, Italy

Université d'Artois, France

China University of Geosciences, China

University of Malaga, Spain

Universidad de la República, Uruguay

University of Cádiz, Spain

Universidad de la República, Uruguay

Politecnico di Milano, Italy

University of Hamburg, Germany

Mark Little
Massimo Mecella
Giovanni Quattrocchi
Wolfgang Reisig
Norbert Ritter
Damina Tamburri
Erik Wittern

Red Hat, UK
SAPIENZA Università di Roma, Italy
Politecnico di Milano, Italy
Humboldt-University Berlin, Germany
University of Hamburg, Germany
DEIB Politecnico di Milano, Italy
IBM T.J. Watson Research Center, USA



BiAgent-Based Model for IoT Applications

Case of a Collision Avoidance System

Souad Marir^(✉), Roumeissa Kitouni, Zakaria Benzadri, and Faiza Belala

LIRE Laboratory, Department of Software Technologies and Information Systems,
University of Constantine 2 Abdelhamid Mehri, Constantine, Algeria
souadmarir94@gmail.com, kitouni.romaissa@gmail.com, benzadri@gmail.com,
faiza.belala@univ-constantine2.dz

Abstract. The Internet of Things (IoT) consists in connecting every aspect of daily and professional life to a common infrastructure, in order to improve considerably the efficiency of otherwise unthinking objects. The huge scale on which they operate, as well as the lack of adequate standards and infrastructures makes the development of IoT applications a task of gradually growing complexity. The objective of this work is to define a formal model with BiAgents (Bigraphical Agents) for IoT applications, based on a suggested generic multi-layered architecture. We show how bigraphs support the structural aspects modelisation of these applications while the agents specify their analytical and decisional aspects. We proceed then to the edition and execution of our model using the bigraph implementation tool (RCTool4Bigraphs), and through the exploitation of its model-checker, we formally verify its most critical property. As a practical example, we study the case of a Collision Avoidance System.

Keywords: Advanced driver assistance systems · BAM4IoT
Bigraphs · BiAgents · Collision Avoidance System
Formal Specification · Internet of Things · RCTool4Bigraphs

1 Introduction

The Internet of Things (IoT) is the vision of a world where each entity has a physical or virtual representation, as well as a presence on existing or future interoperable networks. These entities interact through specific protocols in order to offer and consume services. They can generally perceive their environment and affect it.

The development of IoT applications becomes each year more complex and challenging. This is due to, among others, the need of ensuring interconnectivity and supporting a huge scale of interconnected devices. An important point to consider is that this type of application relies to a great extent on shared infrastructure and protocols (the Internet in the most common case). A poorly

designed application may thus not only result in poor behaviour, but also in damaging the infrastructure or hindering other applications. To fully exploit the widely recognized mathematical formalism potential in analysing, designing and implementing IoT complex systems (with which human users and different devices interact), well-defined development approaches are required. To address this issue and specifically to model the IoT applications at different levels of abstraction, a new incremental approach is proposed. In fact, to date in literature, several approaches for the development of IoT applications have been proposed. It is possible to recognise some operational approaches based on multi-agent frameworks [1] to support the implementation of IoT systems. Nevertheless, no well-formalized approach able to support analysis, design and implementation phase of IoT applications development is currently available. In this paper, we address such issue by suggesting a systematic comprehensive approach.

In the beginning, we give a generic layered architecture for IoT applications allowing a separation of concerns mastering thus their complexity. Then, this architecture constitutes an intermediate model for the formal one. This latter is based on a judicious combination of bigraph model [2] and agents, allowing to describe either aspects of an IoT application with precision as well as a high level of abstraction. The defined model may support both IoT system's structure and its behaviour. Indeed, Bigraphs give a way to represent structural aspects according to two axis: locality (place graph) and connectivity (link graph). In addition, a mechanism is provided to express the evolution of Bigraphs, called the Reaction Rules. This makes it possible to describe the behaviour and the states of a dynamically evolving system. Aside from the two previously mentioned aspects of structure and behaviour, IoT applications are also characterized by cognitive aspects, like awareness of the environment and the ability to affect it. Intelligent agents may maintain these aspects, while providing an ideal tool for the specification of communication and context aware interactions. The paper contribution is twofold, on one hand we propose a formal approach for IoT applications in order to master their complexity, on the other hand we extend the BiAgents definition given in [3] for the modelling of this kind of systems.

The rest of this paper is structured as follows. In Sect. 2, we make a summary of the useful concepts for the comprehension of this work's main contribution. Section 3 introduces our multi-layered generic architecture, as well as our case study, the Collision Avoidance System. Section 4 presents our proposed BiAgent-based model for IoT applications while illustrating it with a realistic example. In Sect. 5, we establish a synthesis of related works that use formal methods to model IoT applications. Finally, conclusion and future direction of this work are drawn.

2 Basic Concepts

This section introduces the different formal concepts we use to model the IoT.

2.1 Bigraphs

In Bigraphs theory [2], a *bigraph structure* facilitates the understanding and design of complex systems. Its formal notation guarantees a safety regarding the correction of the modelling. In addition, the *reaction rules* clearly specify the dynamics of the modelled system.

A bigraph structure is the combination of two graphs (see Fig. 1):

- *The place graph* is in the form of a forest of trees each having a root called a “region”. It has a control function that assigns each node a control¹. It can also contain sites that are abstractions in which we can insert other bigraphs.
- *The link graph* is used to represent relationships and connectivity in a system. It has the structure of a hypergraph.

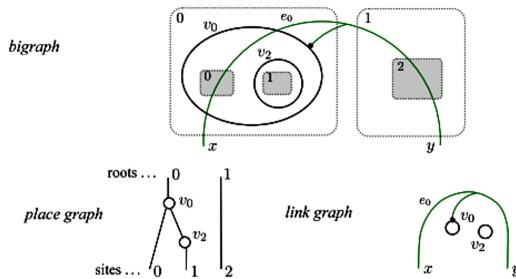


Fig. 1. Example of a bigraph [2]

Definition 1. A *bigraph* [2] is a tuple of form: $G = (V, E, ctrl, prnt, link) : \langle m, X \rangle \rightarrow \langle n, Y \rangle$ where $\langle m, X \rangle$ and $\langle n, Y \rangle$ are the inner and outer faces of G . V and E are respectively, a final set of nodes and a final set of hyperedges. $ctrl : V \rightarrow \mathcal{K}$ is the control function which assigns a control to each node. $prnt$ represents the parent function. $link$ is the function that represents the different links contained in the bigraph.

A reaction rule [4] has the form $R \rightarrow R'$ where R is known as the bigraph redex and R' the reactum one. If we write $B \rightarrow B'$, this means that there is a reaction rule that can be applied to B and give B' , conversely if we write $B \not\rightarrow B'$, this means that there is no possible reaction rule for B to get B' .

Definition 2. A *Bigraphical Reactive System (BRS)* is defined by the set of bigraphs representing the different states of the system obtained from the initial bigraph, and the set of reaction rules applied successively.

¹ A number of ports.

2.2 BiAgents

Biographical Agents [1] are defined by a *physical structure* and a *logical structure*. The physical structure is modelled by the formalism of *bigraphs* and the logical one by *agents*.

Definition 3. *The physical structure* [3] of a BiAgent is defined as a tuple $\mathcal{B} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$ where \mathbb{B} is the space of bigraphs, \mathcal{R} is the set of reaction rules, \mathbb{U} is the control space such as $\mathbb{U} \subseteq \mathcal{R} \times V_{\mathbb{B}}$, with $V_{\mathbb{B}}$ the set of bigraph nodes. B_0 is the initial bigraph. Before defining F , dec3 is a function which gives a valid decomposition of a bigraph into three bigraphs such as $\text{dec3}(B) = (B', B'', B''')$ and B is the composition $B' \circ B'' \circ B'''$. V_R is the set of redex's nodes and $V_{R'}$ is the set of reactum's rules. F is the transition function that, from the current bigraph and from a control action, gives a new bigraph: $F : \mathbb{B} \times \mathbb{U} \rightarrow \mathbb{B}$. It is defined as follows by considering C as the context of R' and d the parameters of $R' : F(B, (R \rightarrow R', h)) = C \circ R' \circ d$ if $\exists \text{ dec3}$ such as $\text{dec3}(B) = (C, R, d)$ and $h \in V_R$ and $h \in V_{R'}$. If not, F is undefined.

Definition 4. *The logical structure* [3] of a BiAgent is defined as a tuple $a = (\mathcal{O}, \mathcal{U}, \text{host}_0, \text{obs}, \text{ctr}, \text{mgrt})$ where \mathcal{O} is the agent's observation space such as $\mathcal{O} \subseteq \mathbb{B}$. \mathcal{U} is the control space such as $\mathcal{U} = \mathcal{R}_a \times V_{\mathbb{B}}$. $\text{host}_0 \in V_{\mathbb{B}}$ is the node that hosts the agent initially. obs is the observation function, $\text{obs} : \mathbb{B} \times V_{\mathbb{B}} \rightarrow \mathcal{O}$. ctr is the control function with which an action can be executed, $\text{ctr} : \mathcal{O} \rightarrow \mathcal{U}$. mgrt is the migration function that, with the host of the agent and an observation, provides the next host, $\text{mgrt} : V_{\mathbb{B}} \times \mathcal{O} \rightarrow V_{\mathbb{B}}$.

2.3 Trace

Definition 5. In a Biographical Reactive System, a trace [5] is a sequence of bigraphs $\langle a_1, a_2, \dots \rangle$ such that for each a_i and a_{i+1} , There is a reaction rule $a_i \rightarrow a_{i+1}$. If there are two traces s and t and the last element of s is the redex of a reaction rule whose reactum is the first element of t , the composite trace exists and begins with all the elements of s followed by all the elements of t , in this case t is an extension of s . We denote $\text{Tr}(A)$ the set of all traces for a given BRS A .

We use traces to make a history of the different events that happen in the system modelled. We can note $t = B_0 \xleftarrow{R1} B_1 \xleftarrow{R2} B_2 \xleftarrow{R3} B_3$ with $R1, R2$ and $R3$ the reaction rules that allow the transition from a bigraph to another. On this basis, a BiAgent can be an agent with a memory by using, as an observation space, a set of traces of bigraphs instead of using a set of bigraphs. If we have a system that contains x agents, we can have x traces, each one representing the course of actions of a particular agent; we call each of these traces a projection.

Definition 6. We represent the flow of agents in space through a projection t^a of a trace t defined as follows:

$$t^a = (p_0^a, h_0^a) \prec (p_1^a, h_1^a) \prec \dots$$

where each p_i^a is a projection of the bigraph B_i Which holds only the node in which the host is located h_i^a .

3 A Layered-Architecture for IoT Applications

Nowadays, we are witnessing a radical evolution of the current Internet in a network of interconnected objects that not only collects information from the environment (detection) and interacts with the physical world (action/control), but also uses existing Internet standards to provide services for information transfer, analytic, applications and communications [6]. In this context, an IoT application is the set of software and hardware that enables a smart behaviour from an ordinary object. In other words, it is a functional system whose aim is to collect data, process it and emit some output that is relevant to its intended task. Specifically, we can summarize the main characteristics of IoT applications as follows [7,8]: *Interconnectivity, Heterogeneity, Dynamic changes, Scale, Security and Connectivity*.

In the present work, the software part of an IoT application will be our main focus; we give a multi-levels architecture of the Connected Objects applications and discuss its implication for the objects communications in terms of the traffic that will be generated. Figure 2 shows the given architecture knowing that an IoT application could be specified according to four layers:

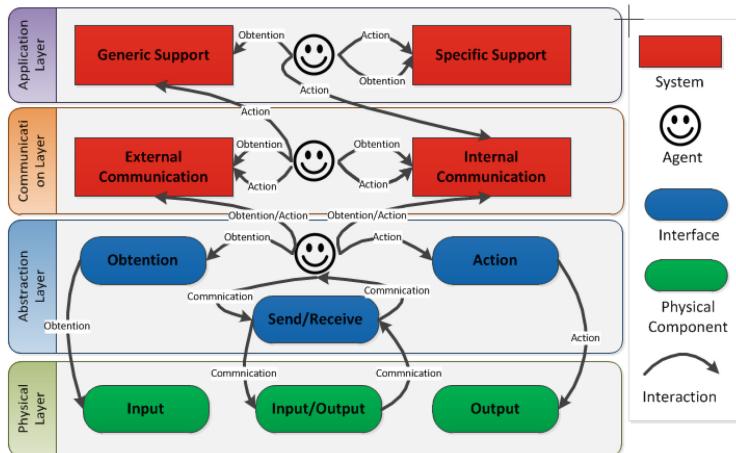


Fig. 2. Multi-layered architecture for an IoT application

Physical Layer. It represents the system's hardware. In particular, we focus our interest on input and output hardware, as they are the mean to interact with and to receive signals from the environment.

Abstraction Layer. It is a low-level layer which implements all interaction protocols with the components within the Physical Layer (in the form of implementable driver modules). In addition, this layer offers an interface through which the application interacts with the physical components. The main reason we choose to represent this layer is the diversity of the used hardware for this kind of systems as well as the lack of standards. We need this layer to

be provided with basic intelligence to analyse input at a low level and decide where to send it; we represent this by an agent that manages this particular layer: the Abstraction agent.

Communication Layer. Since communication is an essential part of every IoT system, we choose to dedicate a separate layer to it. This layer is divided into two sublayers: Internal Communication (ICL), which is the set of protocols used for interaction between all layers within a system, and External Communication (ECL), which is the set of protocols used to communicate with separate, external systems. The intelligent entity managing this layer is the Communication agent, which verifies the correctness of package formats and sends them to the right destination.

Application Layer. This most high-level layer is the actual IoT application. It is again divided into two sublayers: Generic Support for all non-functional aspects which are common to a large spectrum of applications, and Specific Support which is the handling of the most particular aspects of a system. Decisions on the global behaviour of the application are ultimately issued by this layer's smart entity: the Application agent.

Each of the previously mentioned agents are conceived according to a control-loop model as shown in Fig. 3.



Fig. 3. Principle of an agent control loop

Illustrative Example

The case we will study in this section is that of a Collision Avoidance System (CAS) [9] as implemented in a car. It can detect a stationary natural obstacle and issue a warning to the driver and to an external server so as to warn those who might take the same road. It can also detect a moving obstacle (like a pedestrian) or receive a warning from an external server; in either of these cases, it will simply produce a warning for the driver.

By applying the proposed multi-layered architecture (of Fig. 2) to the CAS example of this section, we obtain the structure in Fig. 4;

- In the Physical Layer, one radar and two cameras (one in the front, one in the rear) are used to get raw information on the environment. A vibrator and speakers are used to issue warnings to the driver. A network card is used to communicate with other similar systems through a trusted server. To each kind of these components, an interface in the Abstraction Layer is associated.
- In the Communication Layer, we specify the subsystems that constitute the two sublayers. The verification subsystem in the ECL checks whether a received package is from a trusted server or needs to be destroyed.

- The Application layer was extended with all functional and non-functional high-level software components that specify the actual application behaviour.

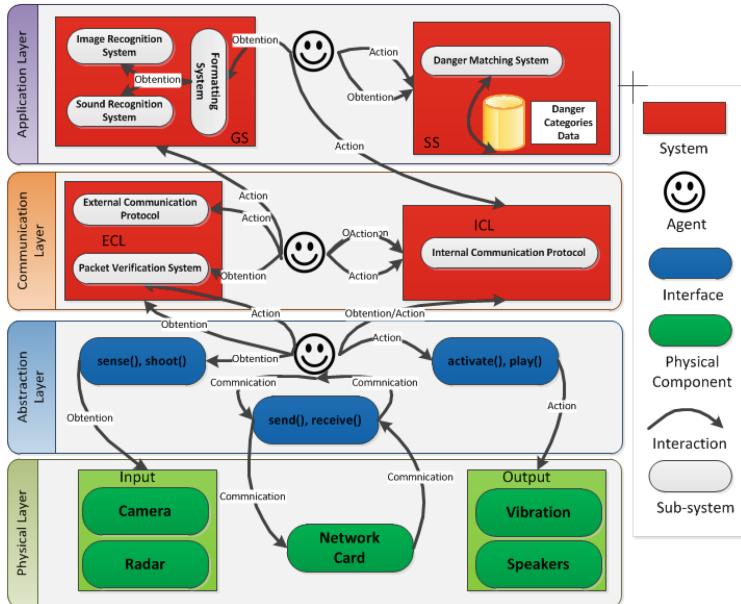


Fig. 4. Multi-layered architecture applied to the Collision Avoidance System

The major components of our proposed architecture for CAS example are used for an easy management and development of this application by both users and programmers. The re-usability and genericity of this approach allow the designer to generate models for any IoT application. In the following we describe the choices that have been taken to satisfy these requirements; we are interested by the formal description and verification issues. Particularly, we show how we generate a formal definition of IoT applications, based on the theory of bigraphical

4 BAM4IoT: A BiAgent-Based Model for IoT

The generic architecture presented earlier represents an intermediate phase towards the development of an appropriate formal model based on BiAgents [3]. Indeed, our model may have two distinct but complementary views: the physical view (the layers) that define the place the connectivity of components of an IoT application and the logical view (the agents) that manages these layers by obtaining information, analysing it and take an action according to the given information. We detail in the following sections the proposed Bigraphical

Agents Model for the IoT (BAM4IoT) according to its two structures (*physical structure* and *logical structure*).

Definition 7 (BAM4IoT Physical Structure). *The physical structure of BAM4IoT model is defined as following:*

$$\mathcal{B}_{IoT} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

- \mathbb{B} is the set of bigraphs modelling the chosen IoT application.
- \mathcal{R} is the set of reaction rules describing its behaviour.
- $\mathbb{U} \subset V_{\mathbb{B}} \times \mathcal{R}$ is the set of controls representing the potential actions recording to a single node with $V_{\mathbb{B}}$ the set of nodes of every $B \in \mathbb{B}$.
- $B_0 \in \mathbb{B}$ is the bigraph representing the initial state of the IoT system modelled.
- F is the control function defining the transition between a bigraph and another according to a control $u_i \in \mathbb{U}$.

Definition 8 (BAM4IoT Logical Structure). *The logical structure of BAM4IoT model is defined by a set of adaptive agents aIoT, each agent $a(i)IoT$ has the following format:*

$$a_{IoT}^{(i)} = (\mathcal{O}, \mathcal{U}, \mathcal{D}, host_0, obs, an, ctr, mgrt)$$

with:

- $\mathcal{O} \subset \mathbb{B}$ is the observation space.
- \mathcal{U} is the control space which represents the possible agent's actions.
- \mathcal{D} is the decision space obtained after the agent's analysis.
- $host_0$ is the agent's initial host.
- the function of observation obs which provides an observation $o \in \mathcal{O}$ using a bigraph and the host of the observant agent: $obs(b, h) = o$.
- the analysis function that, with an observation or a set of observations and a host, analyses this host or its sons and returns a positive or negative decision: $an(o, h) = \alpha \in \mathcal{D}$.
- The control function which gives the next succession of rules to be executed, each according to a node, using the result of an analysis: $ctr(\alpha) = u \in \mathcal{U}$.
- The migration function that provides the next host of the agent according to the current host and an observation: $mgrt(o, h) = h'$.

CAS Example Application

Let us take again the CAS example (Collision Avoidance System) and try to define their structures (physical and logical), as well as its behaviour while illustrating the BAM4IoT definitions.

The Physical Structure of CAS example is given by the tuple

$$\mathcal{B}_{CAS} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

- $\mathbb{B} = \{B_0, \dots, B_{20}\}$ is the set of all bigraphs resulting from the application of the defined reaction rules.

Table 1. Collision avoidance system reaction rules

Rule ID	Description
R0	A sensor device (radar) determines the presence of an input (pedestrian)
R1	The information is sent to the ICL (Internal Communication Layer) in the purpose of being formatted
R2	The information is formatted according to the Internal Communication Protocols (ICP) and its path is traced
R3	The formatted information is transmitted to the system (GS)
R5	The system looks for internal data on known dangers that may correspond to the extracted information
R6	An action corresponding to the detected danger is requested
R7	The action is transmitted to the Internal Communication System (ICS) to be carried out on the Current System (CS)

- $\mathcal{R} = \{R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$. A summary of these reaction rules is given in Table 1.
- $V_{\mathbb{B}}$ the set of nodes of every $B \in \mathbb{B}$ with $V_{\mathbb{B}} = V_{SS} \uplus V_I \uplus V_F \uplus V_E \uplus V_{ES} \uplus V_S$.
- \mathbb{U} the set of every possible couple made of reaction rules and nodes: $\mathcal{R} \times V_{\mathbb{B}}$.
- $B_0 \in \mathbb{B}$ is the bigraph representing the system's initial state B_{SDC} (Fig. 5).
- F is the control function that defines the transition from a bigraph to another through a particular control. If a rule isn't applicable to the node to which it is associated in the control $u_i \in \mathbb{U}$, F is undefined.

The Logical Structure of CAS example is given by a set of three agents: AppAg, ComAg, AbsAg, each one manages changes of the corresponding root (layer) [10]. For lack of space, we explain only the following agent, which is judged the more relevant (see [10] for more details):

$$App^{Ag} = (\mathcal{O}^{App^{Ag}}, \mathcal{U}^{App^{Ag}}, \mathcal{D}^{App^{Ag}}, host_0^{App^{Ag}}, obs^{App^{Ag}}, an^{App^{Ag}}, ctr^{App^{Ag}}, mgrt^{App^{Ag}})$$

with:

- $\mathcal{O}^{App^{Ag}} = \{B_6, B_8, B_{10}\}$
- $\mathcal{U}^{App^{Ag}} = \{(R5, IE), (R7, IE)\}$
- $\mathcal{D}^{App^{Ag}} = \{\alpha_1^{App}\}$
- $host_0^{App^{Ag}} = 2$
- $obs^{App^{Ag}}(B, h) = \begin{cases} obs_1^{App} & \text{if } B = B_6 \text{ and } h = 2 \\ obs_2^{App} & \text{if } B = B_8 \text{ and } h = FA \\ obs_3^{App} & \text{if } B = B_8 \text{ and } h = FS \\ obs_4^{App} & \text{if } B = B_8 \text{ and } h = GS \\ obs_5^{App} & \text{if } B = B_{10} \text{ and } h = IE \end{cases}$

$$\begin{aligned}
 - an^{App^{Ag}}(o, h) &= \alpha_1^{App} \text{ if } o = \{obs_1^{App}, obs_2^{App}\} \text{ and } h = GS \\
 - ctr^{App^{Ag}}(\alpha) &= \{(R6, IE), (R7, IE)\} \text{ if } \alpha = \alpha_1^{App} \\
 - mgrt^{App^{Ag}}(o, h) &= \begin{cases} FA & \text{if } o = obs_1^{App} \text{ and } h = 2 \\ FS & \text{if } o = obs_2^{App} \text{ and } h = FA \\ GS & \text{if } o = obs_3^{App} \text{ and } h = FS \\ IE & \text{if } o = obs_4^{App} \text{ and } h = GS \\ 2 & \text{if } o = obs_5^{App} \text{ and } h = IE \end{cases}
 \end{aligned}$$

In addition to this definition, we can visualize a system's behaviour through the use of traces and projections. This will illustrate all state changes that happen alternatively to the physical structure (CAS reaction rules) and to the logical structure (an agent's migration). Describing an application through the use of this extended model serves the main purpose of preventing non-deterministic behaviour. As a matter of fact, there is no way to guarantee a consistent execution path with Bigraphical Reactive Systems alone. A reaction rule's applicability is decided through a pattern matching and could be applicable without it being semantically appropriate. The control function on the other hand always associates to a given analysis (of the current state) an applicable and semantically appropriate set of actions. Figure 5 shows a scenario of detection of pedestrian. After applying CAS reaction rules (R_0 to R_7), the system arrives at a final state representing the action performed.

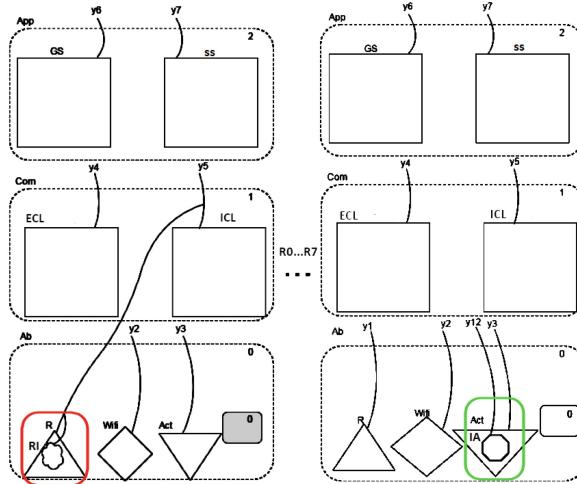


Fig. 5. The BAM4IoT of collision avoidance system B_{CAS}

For validation purpose, we exploit the RCTtool4Bigraphs (an extended and generic version of a tool that has been already proposed in [11] and [12]) to check that our BAM4IoT specification satisfies an important property. Figure 6 shows

```
(built-in equation for symbol modelCheck)
modelCheck(prnt< nIB >=< nR > ; prnt< nQQ >=< rApp > ; prnt< nCCI >=< rCom > ;
prnt< nR >=< rAb > ; prnt< nGS >=< rApp > ; prnt< nCCE >=< rCom > ; prnt<
nWIFI >=< rAb > ; prnt< nAct >=< rAb > ; prnt< s0 >=< rAb > ; link< e5,nR
>=< nCCI > ; link< e1,nR >=< y1 > ; link< e2,nWIFI >=< y2 > ; link< e3,nAct
>=< y3 > ; link< e4,nCCE >=< y4 > ; link< e5,nCCI >=< y5 > ; link< e6,nGG
>=< y6 > ; link< e7,nGS >=< y7 >, False R (True U ActionAttainability))
-->
true
rewrites: 25 in 11889324805ms cpu (22ms real) (0 rewrites/second)
result Bool: true
```

Fig. 6. Model checking results of the Action-Attainability property.

model checking results of the *Action-Attainability* property; we conclude that the entered specification verifies the desired property.

5 Related Work

In their paper about high-level application development in IoT [13], authors mention three main approaches for developing IoT applications: Middleware, Programming framework and Model-Driven Development (MDD). The latter, which is most relevant for our current work, consists of describing the applications as high-level abstract models, and then using these models according to some research ways (code generation, formal analysis, etc.).

Few works in the literature have focused on the use of formal methods to design and specify such applications by supporting the characteristics of complex, distributed and auto-adaptive systems. For instance, authors of [1] based their framework for the IoT on agents, focusing on self-adaptive and self-organizing aspects of the applications. On the other hand, graph-based formalisms are usually favoured to support the description of interrelation as well as dynamic configuration. The work cited in [14] tried to combine the advantages of both formalisms and proposed a hybrid one using complex networks, a graph-based formalism, as well as agent-based modelling for the IoT. The idea of creating a formal model out of many different methods stems from their individual failure to keep up with the ever-growing complexity of modern systems, and the decreasing accuracy with which they can describe IoT applications.

In the same thought, our proposed approach combines: (1) Bigraphs to describe geographical dispersion, connectivity and dynamic changes; and (2) Agents to model communications and context-aware interactions in IoT applications. In particular, the proposed formal model (BAM⁴IoT) gives an unified way to describe IoT applications so as to result in a deterministic, understandable specification. Moreover, we used a practical tool (RCTool4Bigraphs [11, 12]) to verify an IoT-related property (*Action-Attainability property*).

6 Conclusion

A recurring challenge in the development of IoT applications is managing their complexity. To overcome this, we have set ourselves the main objective of applying one of the most appropriate mathematical formalisms. Our attention has turned to the bigraphs and the agents. A judicious combination of these two formalisms allowed to consider the two aspects inherent to this type of applications; the associated *physical object/virtual intelligence* pair.

We have chosen to develop first, a layered architecture to simplify the description of IoT applications and its instantiation according to a case study: Collision Avoidance System (CAS). Then, we were able to validate the physical structure of the extended formal model BAM4IoT with a model-checking tool of bigraphs (RCTool4Bigraphs).

We propose, in the future works, to give our model a way for objects (agents) to communicate by extending the definition of the agents in BAM4IoT with a suitable function. Obviously, a tool around the bigraphs that supports the virtual or logical aspect of the systems is an unavoidable solution.

References

1. do Nascimento, N.M.: Fiot: an agent-based framework for self-adaptive and self-organizing applications based on the internet of things. *Inf. Sci.* **378**, 161–176 (2017)
2. Milner, R.: *The Space and Motion of Communicating Agents*. Cambridge University Press, Cambridge (2009)
3. Pereira, E., Kirsch, C., Sengupta, R.: Biagents-a bigraphical agent model for structure-aware computation. *Cyber-Physical Cloud Computing Working Papers*, CPCC Berkeley (2012)
4. Perrone, G.: Domain-Specific Modelling Languages in Bigraphs. Ph.D. thesis, IT University of Copenhagen Copenhagen, Denmark (2013)
5. Perrone, G., Debois, S., Hildebrandt, T.: Bigraphical refinement. arXiv preprint [arXiv:1106.4091](https://arxiv.org/abs/1106.4091) (2011)
6. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013)
7. Patel, K., Patel, S.: Internet of Things-IOT: definition, characteristics, architecture, enabling technologies. application & future challenges. *IJES&C* **103**, 62–84 (2016)
8. Vermesan, O., Friess, P.: *Internet of Things - From Research and Innovation to Market Deployment*. River Publishers, Aalborg (2014)
9. Advanced Driver Assistance Systems (ADAS) — TOSHIBA Storage & Electronic Devices Solutions Company (2017)
10. Marir, S., Kitouni, R.: Combinaison des agents et des bigraphes pour la modélisation des applications iot. Master's thesis, Université Constantine 2 Abdelhamid Mehri (2017)
11. Benzadri, Z.: Spécification et Vérification Formelle des Systèmes Cloud. Thesis, Université Constantine 2 - Abdelhamid Mehri, October 2016

12. Benzadri, Z., Bouanaka, C., Belala, F.: Big-CAF: a biographical-generic cloud architecture framework. *Int. J. Grid Utility Comput.* **8**, 222–240 (2016)
13. Patel, P., Cassou, D.: Enabling high-level application development for the internet of things. *J. Syst. Softw.* **103**, 62–84 (2015)
14. Batoon, K., Niazi, M.A.: Modeling the internet of things: a hybrid modeling approach using complex networks and agent-based models. *Complex Adapt. Syst. Model.* **5**(1), 4 (2017)



Seamless Interactions on the Internet of Things. A Spotify-Based Proof of Concept

Jose Garcia-Alonso¹(✉), Javier Berrocal¹, Carlos Canal², and Juan M. Murillo¹

¹ University of Extremadura, Caceres, Spain

{jgaralo,jberolm,juanmamu}@unex.es

² University of Málaga, Málaga, Spain

canal@lcc.uma.es

Abstract. As the number of devices connected to the Internet of Things increases, so increases the interactions required between users and those devices and systems. In a world where non-technically inclined users live surrounded by Internet of Things systems, the barriers to entry for the use of these technologies should be as low as possible. In these circumstances, the Situational-Context is a new computational model to allow Internet of Things software to automatically adjust its behaviour to the context of its users. In this paper we present a Spotify-based proof of concept of the Situational-Context. The users of this system can seamlessly agree on the music played in a public environment without direct interaction between them or with the system. With this proof of concept we address some of the main challenges raised by the implementation of the Situational-Context as well as demonstrate the benefits it provides to Web of Things systems in terms of simplified user interaction and improved context adaptation.

Keywords: Internet of Things · Context-Aware · Interactions

1 Introduction

The capabilities of embedded devices keep increasing every year, leading to the development of a huge variety of smart things. These devices are connected to the Internet and provide a virtual representation of themselves. Other devices can interact with these virtual representations creating the Internet of Things (IoT) [11]. One of the main goals of this paradigm is to simplify people life by making the technology work for them, either providing more information for decision-making or facilitating the accomplishment of some tasks.

Predictions estimate that, by 2020, there will be 50 to 100 billion of these devices connected to the Internet [24]. However, if we analyse the current state of how people interact with them, the benefits provided will not be as groundbreaking as expected. With the aim of increasing the usability of smart things, the

behaviour of these systems usually depends on the users preferences and their context, which can shift considerably over time. Therefore, manually configuring an increasing number of smart things connected to daily life activities needs too much attention. Specially, taking into account that every change of the users' context or preferences needs a reconfiguration of the smart devices. This manual control of IoT systems, that is acceptable when working with a small number of devices, will became a burden for users involved in dozens of systems.

Accordingly, solutions are needed to transparently and effortlessly integrate people's needs, moods and preferences into the connected world of the IoT. In the literature we can find research works dealing with gathering and processing the contextual information of users in order to create more comprehensive virtual profiles [10]. Nevertheless, an accurate and comprehensive virtual profile is not enough. Techniques to adapt the system behaviour to the context are also necessary. Currently, researchers are working on techniques such as Dynamic Composition [5] or Context Oriented Programming (COP) [14]. These techniques allow developers to predefine different behaviour of an application depending on the identification of specific contextual information. They concur that the information and/or variables triggering the adaptations is detailed within the source code of the applications. Thus, the adaptation capabilities are limited to the set of predefined contexts. However, the variability of this information is very large and is difficult to identify and express every plausible situation in the development phase.

To address this situation, we have proposed the concept of Situational-Context [2] as a way to analyse the conditions that exist at a particular time and place; and how this analysis can be used to predict, at run-time, the expected behaviour of IoT systems. In this paper, we present a first implementation of the Situational-Context model. Specifically, a Spotify-based application that allows users to seamlessly agree on the music played in a public environment without direct interaction between them or with the system. This proof of concept allows us to showcase the main benefits provided by the Situational-Context.

The rest of the paper is structured as follows. Section 2 presents the motivations that lead to the development of this work and details the Situational-Context model to provide a clear view of its characteristics. Section 3 describes the developed application focusing on the benefits provided by the Situational-Context model. Section 4 lists the most relevant works related to the one present here. Finally, Sect. 5 presents some conclusions and future works.

2 Situational Context

In the IoT usually several devices are orchestrated to build complex systems [17]. However, as the IoT is more integrated into people daily activities this orchestration becomes more complex, since systems should dynamically adjust to the needs of each individual and the specific circumstances of each moment and place.

In this sense, there are different researchers focused on the identification of people's context and its use to adjust applications behaviour. So far, this context

was manually set by the user. With the aim of improving the user experience and reducing the effort required to set these preferences, different approaches, such as Ambient Intelligence [6] and Context-Aware [15], have been defined to semi-automatically identify them.

In the last few years, the increased computing and storage capabilities of mobile devices, allowed the authors of this paper to propose a new context-aware computing model, named People as a Service (PeaaS) [12]. This model uses the user's mobile device to gather, store and compute the contextual information in order to construct his/her virtual profile. Thus, in the orchestration of smart things, profiles can be used to make better adaptations. In a further step, the Internet of People (IoP) [20] propose an infrastructure and a manifesto for IoT systems that support this proactive adaptations. Recent research also allowed us to prove the situations under which these models also reduces the consumption of mobile resources [3].

PeaaS defines that a profile is divided into at least two different facets. A *Basic Profile* containing the dated raw contextual information with the user's status, the relationships with other devices and its history. And a *Social Profile*, which contains the results of high level inferences performed over the Basic Profile.

A specific characteristic of this computing model is that the virtual profile stored in the mobile device can also be provided as a service to other applications, to third-party companies or to other devices. Thus, the surrounding devices can reuse it to adapt themselves to the user's preferences.

One of the main goals of inferring high level information is to better adapt the applications and the surrounding devices' behaviour to the users. Currently, there are different paradigms proposing languages and frameworks to define the situations under which the applications or the devices can be adapted and how this adaptation should be. For instance, the Context Oriented Programming paradigm provides an additional dimension to standard programming techniques to dynamically switch among the behaviours associated with each context, such as bandwidth availability, WiFi connection, battery level, etc. [26]. The Dynamic Composition paradigm goes a step forward when the interactions between devices cannot be identified at the development stages. It allows developers to implement the application behaviour and interactions without defining the specific devices involved [13].

Therefore, currently there are a lot of proposals for building more comprehensive virtual profiles, better detailing the needs of each individual, her status, personality, desires, etc. However, the techniques for developing systems adaptable to the users' profiles requires to predefine in the development phase the relationships between contextual data and the different behaviours or actions that can be triggered. This limits the customization of applications and makes it difficult to obtain IoT systems totally responsive to the users' virtual profiles. In addition, in IoT environments, the number of devices with which a user can interacts with, the frequency at which these interactions occur and, above all, the social aspect of these interactions raise the need of paradigms focused on

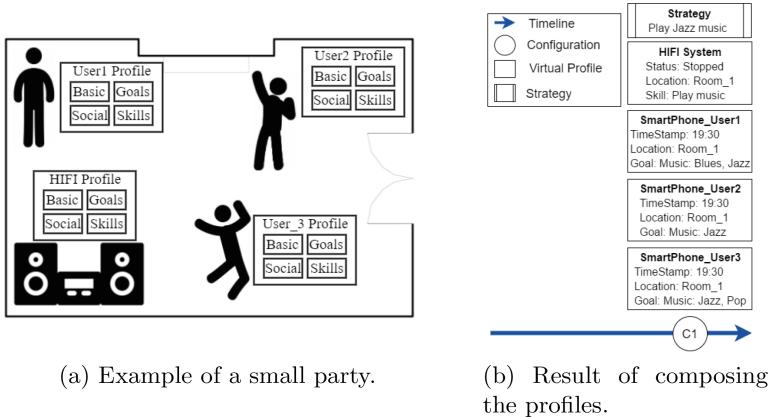
identifying at run-time the behaviours and interactions that should emerge from the concrete situations and how the system should be able to respond in an ad-hoc way to them.

Recently, we have been working on a new concept called Situational-Context [2]. The Situational-Context is a way to analyse the conditions that exist at a particular time and place in order to predict, at run-time, the expected behaviour of IoT systems. To that end, first, the Situational-Context adds two new subsets of information to the virtual profile defined in PeaaS. The *Goals* detailing the status of the environment desired by the entity. And the *Skills* or capabilities that an entity is endowed with, in order perform actions capable of modifying the environment and aimed at achieving Goals.

Taking into consideration that in most IoT systems there are different entities (things and people). And that each of them has its own virtual profile. We define the Situational-Context as the composition of the virtual profiles of all the entities involved in a particular situation. The result of composing the virtual profiles is the combined history of the entities ordered in a single timeline, the result of high level inferences performed over the combined virtual profiles, the set of Goals of the entities and their Skills. From the combined information of the Situational-Context, strategies to achieve Goals based on the present Skills are identified. These strategies will guide the prediction of the interactions that must emerge from the context. Therefore, the Situational-Context provides a higher level of automation of smart things with people.

Furthermore, the Situational-Context is a dynamic abstraction of the combined profiles of the entities involved in a situation and, therefore, evolves and changes through time. To analyse the instantaneity of this context, we use the concept of *Configuration*. A Configuration is the unified and stable view of the virtual profiles of the devices involved in the situation at a specific point in time. When changes in the environment happen, the Configuration is no longer stable and must be updated. Thus, a new Configuration must be defined from the updated/new virtual profiles of the devices. Thus, the Situational-Context can also be seen as a succession of Configurations.

In order to better explain this concept, let us use a simple example of a living room in which there is a small party with several people and a smart HIFI system (Fig. 1a). All of them, through their mobile device, have a virtual profile containing a Basic Profile (with, for example, information about the music they usually play, their location, etc.), a Social Profile (with their musical interests in each specific location) and their Goals (e.g. to listen specific music styles). The smart HIFI system has its own virtual profile with a Basic Profile (containing, for example, the music it has played), a Social Profile (with the music styles that are usually played), the Goals (e.g. to save energy) and its Skills (e.g. to play music). The Situational-Context would be the composition of the virtual profiles, as Fig. 1b shows. From this composed profile, the strategy with the actions that the entities should execute to satisfy the goals are identified. Concretely, the music style and, even, the concrete songs that the HIFI system should play would be detected.

**Fig. 1.** Example of the Situational-Context.

However, this concept has some open challenges: which device or devices should compute the Situational-Context? How the contextual information can be exchanged? How common Goals should be agreed for a specific configuration when the devices involved have different or, even, opposed Goals? How the strategies to achieve Goals should be identified? The following section details a proof of concept for the Situational-Context that provides a possible solution for each of the above challenges.

3 Spot&Joy: A Spotify-Based Situational-Context Application

In this section we describe the Spot&Joy application. This application allows its users to seamlessly control the music played in public environments and it was developed following the Situational-Context model. The application is designed to be used in different scenarios like a small private party in a house, a waiting room with ambient music, or a night club. Figure 2 shows the architecture of Spot&Joy.

As can be seen in the figure, the Spot&Joy application is formed by two main components: the user component and the player component. The user component is in charge of choosing the music based on each user preferences and context and is run by the users smartphones. The player component is in charge of playing the music chosen by the users and, therefore, it must be executed by a device with the skill to play music. Different devices could execute this component, for example a smartphone connected to a set of Bluetooth speakers, a professional set-top box or a smart speaker connected to the internet. In the next sections we describe in detail each of this components.

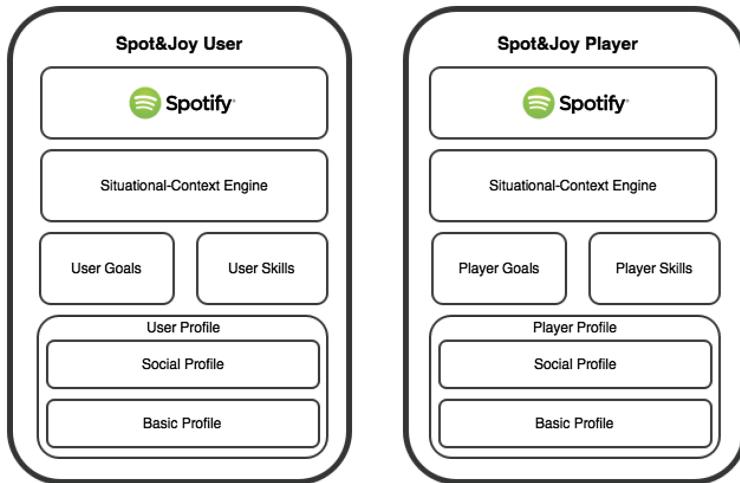


Fig. 2. Spot&Joy architecture.

3.1 Spot&Joy User

The Spot&Joy user component, developed as an Android application, allows users to choose the music that would be played in public environments. To perform this task the system uses the well known Spotify music-streaming service. In particular its Web API [1].

The first time the application is run by a user she must provide her Spotify credentials to be able to use the API. Once the user is logged in Spotify she can use the application as a normal Spotify client, listening to her playlists, searching for different kinds of music, etc.

Additionally, under the Spotify client, the Spot&Joy application integrates several Situational-Context features. Simply by installing the application, the device in which is installed acquires the goal to control the music played in Situational-Context enabled public environments. This goal is registered in the User Goals component of the architecture. This means that, whenever the users find herself in a situation where there is a device with the skill to play music, the played music will be affected by the preferences of the user. The task of detecting the situations in which the goal can be satisfied is performed by the Situational-Context Engine module.

In order to gather and process the context and preferences of the user, the Spot&Joy application integrates a PeaaS implementation [12], called nimBees. NimBees [21] is a commercial mobile push notification platform composed by an API for mobile applications, which stores the users' profile and allows applications to receive segmented push notifications, and a backend, which manages the sent notifications. Once the push notification is sent and reaches the mobile device, the API decides if the owner is an appropriate recipient for that message. These notifications are used to detect the different devices and to exchange the

musical preferences among them. For this proof of concept, the basic and social profile used for the computation of the Situational-Context is limited to the musical preferences of the users, stored in the form of Spotify playlists, and the basic contextual information associated to each playlist playback (geographical position and moment in time).

With this information the Spot&Joy detects when the user is in a situation where the music can be controlled, by the presence of a device with the skill to play music, and uses the profile of the user built by PeaaS to propose the playlist that better adapts to the user context. The Spot&Joy player receives the proposed playlist, computes all the users' playlists and determines the music that would be played at every moment.

3.2 Spot&Joy Player

The Spot&Joy player plays the music chosen by the users. To perform this task the system also uses the Spotify API included in the architecture and, therefore, the user that control the Spot&Joy player needs to provide a valid Spotify credentials.

Once the system has access to Spotify its management is delegated to the users. The device running the Spot&Joy player would expose the skill to play music in the Device Skills component of the architecture. This skill would be detected by the smartphones that have installed the Spot&Joy user application. Each of these users would send their preferred playlist to the Situational-Context Engine component of the player device. The communications between the user devices and the player devices are performed directly, without depending on an external server.

As detailed in Sect. 2, the Situational-Context changes through time. In the case of the Spot&Joy player it changes whenever a user sends a playlist. When that happens a new *Configuration* is computed and it controls the music played until the *Configuration* is changed by the presence of a new user or by the absence of one of the previous users.

For this proof of concept, the computation of a stable *Configuration* of the Situational-Context is relatively simple and satisfy the following rules:

1. If there is no user present in a Situation there would be no music played.
2. When the first user joins the system, her suggested playlist would be played completely and then the Spotify recommendation system would be used to play similar music.
3. When an additional user joins the system, the songs contained in her suggested playlist would be added to a pool containing all the songs suggested by the users.
4. When a user leaves the system, the songs contained in her suggested playlist would be removed from the pool.
5. Whenever the pool of songs changes, the order in which the songs would be played is reorganized according to the following algorithm.

- (a) The songs in the pool are first prioritized by its number of appearances in the pool.
- (b) Songs with the same number of appearances are prioritized by the time their first appearance has been in the pool.
- (c) Songs added to the pool by a user would be penalized when a song suggested by that same user is played.
- (d) Whenever a song is played all its appearances would be removed from the pool.
- (e) If the pool is empty and there are users in the system, the previously played songs would be used as seeds for the Spotify recommendation system.

By following these rules the Spot&Joy player provides a shared playlist that tries to satisfy all the users involved in the system. At the same time, the use of the Spot&Joy application allow users to seamlessly affect the system behaviour to better satisfy their musical preferences and taking their context into account.

3.3 Situational-Context Challenges

As mentioned in Sect. 2, the Situational-Context concept raises some open challenges. The development of the proof of concept presented in this work has allowed us to address them. In this section we describe the adopted solution for each challenge as well as some insights for future developments under the Situational-Context model.

Which Device or Devices Should Compute the Situational-Context?

There are several options to address this challenge. Computations can be performed in one of the devices physically present in the situation, but they can also be performed in a remote server or using a mixed approach. For the Spot&Joy proof of concept we decide to compute the Situational-Context in a physically present device for several reasons. First, because the smart devices computational capabilities are more than enough to compute the Situational-Context. Second, because by avoiding the use of an external computational entity the system becomes more reliable. And third, because it simplifies the development process since there is no need to develop an additional component for the remote entity.

This approach, however, it is not optimal for every situation. For developing an application following the Situational-Context model several factors must be taken into account before deciding where the Situational-Context must be computed. These factors mainly revolve around the resources consumption in the present devices, like battery or data traffic, versus the resource consumption in remote computational entities. A detailed study about the best approach to be used for developing these kind of application was presented in [3].

How Contextual Information can be Exchanged?

For the implementation of the proof of concept presented here we used an ad-hoc solution. Spotify playlists and sets of songs are exchanged between the different entities using the communications mechanisms present in the PeaaS implementation, specifically the use of push notifications. These messages contain the identifiers assigned by Spotify to each playlist and song.

While this solution may be enough for a proof of concept, is far from a generic solution that can be used in the implementation of the Situational-Context model. More research is needed in order to identify the best approach to share contextual information between the entities present in a situation.

How Common Goals Should be Agreed for a Specific Configuration when the Devices Involved have Different or, Even, Opposed Goals?

The rules used by the Spot&Joy player in order to negotiate with the users the music to be played were detailed in Sect. 3.2. Again, this is an ad-hoc algorithm for this proof of concept. However, this algorithm can be generalized as a negotiation strategy that can be used in other implementations of the Situational-Context model.

Specifically, the implemented strategy can be used in any situations where users have a set of tasks that should be processed by a specific device. This task could be songs to play, as in Spot&Joy, but they can also be photos to be shown in a digital frame, computational tasks that have to be executed in a more powerful device, etc. In order to process these tasks, the device processing them needs three elements. A pool or queue of tasks where the tasks sent by users and their details are stored, a prioritization strategy that classifies the tasks by the order in which they should be executed and a user management mechanism that reorders the tasks pool whenever a user enters or leaves the situation. With these elements the strategy can be reused in different Situational-Context computations.

How the Strategies to Achieve Goals should be Identified?

As mentioned in the previous point, the strategy for the Spot&Joy player was created ad-hoc. The idea behind it was to take into account the musical preferences of each user.

As the Situational-Context model matures, more general strategies to achieve goals would be available. And, therefore, a system would be needed to identify the most efficient strategies for each goal and situation. More research is needed regarding this challenge. However, we are developing an initial approach based on the users feedback. When more than one strategy is available to achieve a goal, the users would be asked to rate the success of the strategy in accomplishing such goal. This feedback can be provided manually but can also be gathered automatically from their profiles.

4 Related Works

In the last few years, a lot of researchers have been working on computing the raw contextual information of a user in order to make high level context deductions. In [10], the authors indicate that the user context can be expressed as a combination of the user's activity, light conditions, social setting and geographical location. So, they propose a system to gather the user contextual information. This information is sent to a cloud environment to perform high level inferences. Then, it is reused to adapt the interface of an app to the user environment.

There are other works focused on a more social perspective, computing and sharing the contextual information. In [23] the authors propose a system that can automatically recognize the high-level context of the users, i.e. activities, emotions, and relationships with other users. Once computed, this information is shared in a mobile social network so that other users and devices can take it into account.

In [9], the authors focused on improving the Location-Based Services using contextual information. To that end, the authors propose to combine the contextual information of different users (including their tastes, preferences, activities, social interactions, etc.) and point of interest located within a specific geospatial range in order to promote venues that could be interesting for them.

In addition, in the Context-Aware [15], Ubicomp [4], User Modeling [16] and Ambient Intelligence [19] areas, there are others works related with the computing of different contextual information in order to adapt the devices' behaviour.

There are works focused on managing people contextual information with the aim of improving their experience in multi-device systems [7]. In [18] the authors indicate that in order to enable context-awareness for distributed applications, new architectural styles are needed to support the gathering and interpretation of disseminated context attributes. Therefore, they propose a context-aware middleware based on the client-server architecture. Nevertheless, the use of a central server requires a high communication between the devices and the server, both for storing new information and for consulting the virtual profile [3].

In [25] the authors indicate that ubiquitous computing advocates the construction of massively distributed systems that can be deployed in heterogeneous devices. These systems should take into account the users' context, adapting themselves to different situations. Therefore, they propose a middleware facilitating the development of context-aware agents.

In [22] the authors focused on the Ambient Intelligence environment. They indicate that this environment covers a large number of devices and people. So, in order to achieve a better scalability, they have defined a generic middleware layer for transferring contextual information between devices.

Finally, at the business side and more related with the proof of concept presented in this paper, the application Flo Music [8] allows people to create

social playlists. These playlist can be used to sync nearby smartphones in order to define the songs that should be played. They can be played on a smartphone with a speaker plugged in or can be streamed to all smartphones connected to the playlist and played on all of them. Although this application does not automatically identify the songs that should be added to the playlist depending on the users' preferences, it proofs that the industry has a great interest in combining the users' preferences and needs in order to further automate different social activities.

5 Conclusions and Future Work

Current IoT applications can be implemented to have a specific behaviour depending on the users' preferences and context. However, this adaptation is limited to the behaviours defined in the development phase. In the future these applications shall be fully self-adaptive and able to completely change their behaviour, at run-time, to cover the needs of any user or any group of users, and to use the capabilities of the new devices included in the system.

Here, we present the first implementation of the Situational-Context model in the form of a Spotify based proof of concept. The goal of this implementation is to demonstrate how the different challenges presented by the Situational-Context can be addressed and, also, to show the kind of applications that can be built under this model.

However, there is still much research needed around the Situational-Context model. We are focusing our next efforts on generalizing the solutions for the different challenges and analysing the impact of different communication technologies, such as LTE Direct, WIFI-Aware, etc.

Acknowledgments. This work was supported by the Spanish Ministry of Economy, Industry and Competitiveness (TIN2014-53986-REDT, TIN2015-67083-R and TIN2015-69957-R (MINECO/FEDER)), by 4IE project (0045-4IE-4-P) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Department of Economy and Infrastructure of the Government of Extremadura (GR15098), and by the European Regional Development Fund.

References

1. Spotify Web API. <https://developer.spotify.com/web-api/>
2. Berrocal, J., Garcia-Alonso, J., Canal, C., Murillo, J.M.: Situational-Context: a unified view of everything involved at a particular situation. In: Bozzon, A., Cudre-Marou, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 476–483. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_34
3. Berrocal, J., Garcia-Alonso, J., Vicente-Chicote, C., Hernández, J., Mikkonen, T., Canal, C., Murillo, J.: Early analysis of resource consumption patterns in mobile applications. *Pervasive Mob. Comput.* **35**, 32–50 (2016)
4. Caceres, R., Friday, A.: Ubicomp systems at 20: progress, opportunities, and challenges. *IEEE Pervasive Comput.* **1**, 14–21 (2011)

5. Chen, G., Li, M., Kotz, D.: Data-centric middleware for context-aware pervasive computing. *Pervasive Mob. Comput.* **4**(2), 216–253 (2008)
6. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: technologies, applications, and opportunities. *Pervasive Mob. Comput.* **5**(4), 277–298 (2009)
7. Denis, C., Karsenty, L.: Inter-usability of multi-device systems: a conceptual framework. Multiple user interfaces: Cross-platform applications and context-aware interfaces, pp. 373–384 (2004)
8. FLO Music (2017). <http://www.flomusic.com/>
9. Gasparetti, F.: Personalization and context-awareness in social local search: state-of-the-art and future research challenges. *Pervasive Mob. Comput.* **38**, 446–473 (2016)
10. Gronli, T.M., Ghinea, G., Younas, M.: Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. *Pers. Ubiquitous Comput.* **18**(4), 883–894 (2014)
11. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013)
12. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. *IEEE Softw.* **31**(2), 48–53 (2014)
13. Heo, S., Woo, S., Im, J., Kim, D.: IoT-map: IoT mashup application platform for the flexible IoT ecosystem. In: International Conference on the Internet of Things, pp. 163–170. IEEE (2015)
14. Hirschfeld, R., Costanza, P., Nierstrasz, O.: Context-oriented programming. *J. Object Technol.* **7**(3), 125–151 (2008)
15. Hong, J.Y., Suh, E.H., Kim, S.J.: Context-aware systems: a literature review and classification. *Exp. Sys. App.* **36**(4), 8509–8522 (2009)
16. Kobsa, A.: Generic user modeling systems. *User Model. User-Adap. Inter.* **11**(1–2), 49–63 (2001)
17. Kovatsch, M.: CoAP for the web of things: From tiny resource-constrained devices to the web browser. In: ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. ACM, New York, pp. 1495–1504 (2013)
18. Löwe, R., Mandl, P., Weber, M.: Supporting generic context-aware applications for mobile devices. In: IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 97–102, March 2013
19. Marzano, S.: The new everyday: Views on ambient intelligence. 010 Publishers, Rotterdam (2003)
20. Miranda, J., Makitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., Murillo, J.: From the internet of things to the internet of people. *IEEE Internet Comput.* **19**(2), 40–47 (2015)
21. NimBees. <http://www.nimbees.com/>
22. Olaru, A., Florea, A.M., Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. *Mob. Netw. Appl.* **18**(3), 429–443 (2012). <https://doi.org/10.1007/s11036-012-0408-9>
23. Park, H.S., Oh, K., Cho, S.B.: Bayesian network-based high-level context recognition for mobile context sharing in cyber-physical system. *Int. J. Distrib. Sens. Netw.* **7**(1), 650387 (2011)
24. Perera, C., Liu, C.H., Jayawardena, S., Chen, M.: Context-aware computing in the internet of things: A survey on internet of things from industrial market perspective. *CoRR* (2015)

25. Ranganathan, A., Campbell, R.H.: A middleware for context-aware agents in ubiquitous computing environments. In: Endler, M., Schmidt, D. (eds.) *Middleware* 2003. LNCS, vol. 2672, pp. 143–161. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44892-6_8
26. Salvaneschi, G., Ghezzi, C., Pradella, M.: Context-oriented programming: a software engineering perspective. *J. Syst. Softw.* **85**(8), 1801–1817 (2012)



A Feedback-Based Adaptive Service-Oriented Paradigm for the Internet of Things

Yuji Dong^{1,2(✉)}, Kaiyu Wan², and Yong Yue²

¹ University of Liverpool, Liverpool L69 3BX, UK
yiji.dong@liverpool.ac.uk

² Xi'an Jiaotong Liverpool University, Suzhou 215123, China
{kaiyu.wan,yong.yue}@xjtlu.edu.cn

Abstract. With integrating physical devices into digital world, Internet of Things (IoT) have presented tremendous potential in various different application domains such as smart cities, intelligent transportation, smart home, healthcare and industrial automation. However, current IoT solutions and usage scenarios are still very limited because of the difficulty in sensing the context in continuously changing environments and adaptation to the changes accordingly. The complex dynamic interactions between system components and physical environments are a bit challenging especially when there are other concerns such as scalability and heterogeneity. To solve this problem, a novel adaptive service-oriented paradigm is proposed to support IoT from a low-level viewpoint. The paradigm can overcome some disadvantages of REST (Representational State Transfer) architecture style in the IoT. Two classical examples are illustrated using the proposed paradigm by adding an extra constraint based on REST to improve system states verification and enhance the functionality in modelling physical processes.

1 Introduction

IoT are envisioned to integrate the physical world into computer-based systems. Recent years, with the advanced technique development on sensors, networking and data processing etc., IoT have illustrated great potential in various different fields [12]. However, even after decades of research on system aspects of the IoT, developing IoT based systems is still facing many challenges on the high level system requirements like scalability, inter-operability and fault tolerance [19]. Moreover, most of current IoT applications are coping with data collecting and processing without involving many complex physical behaviours, because current IoT solutions and usage scenarios are still very limited in modeling complex behaviours in continuously changing physical environment.

Context adaptation plays an important role in continuously changing physical environment. Recent years, because of the rapid development of mobile computing and big data, there are plenty of context-sensitive data in the IoT systems, therefore the context-awareness in IoT draws a lot of research attention.

For example, there are many investigation on context-awareness in models [25], architectures [7] and middlewares [6]. On the other hand, adaptation is more difficult than context-awareness. It is usually solved by constructing the feedback loop [2] at different abstracting level like architectures [5], behaviour models [4] and frameworks [22].

REST (Representational State Transfer) is a widely used architecture style and also popular in the IoT fields because of its low entry barrier and scalability merits. However, the REST architecture style was particularly designed for distributed hypermedia systems and it sometimes does not fit IoT requirements. In particular, it is difficult for REST to provide complex operations and high level abstraction, while in the IoT systems, the physical behaviours usually need complex behaviour models which REST cannot provide. Therefore two main issues arise, i.e., system states verification and physical behaviours implementation, which we will discuss in more details in the Sect. 2.

To address these two issues, we propose the Feedback-based Adaptive Service-Oriented Paradigm (FASOP) which can apply at the programming language level to support context adaptation in the IoT systems. Furthermore, the FASOP can be used to add more constraints in order to use the REST style in the IoT systems to overcome these two limitations.

The rest of the paper is organised as follow. Section 2 explains the motivation of the proposed approach. Then the definition and description of the FASOP is presented in Sect. 3. In Sect. 4, the FASOP is applied in the REST as an extra constraint. Section 5 illustrates a simple implementation of the FASOP and the two cases discussed before are implemented with the FASOP to express the advantages in Sect. 6. Finally, Sect. 7 compares some related works and Sect. 8 gives the conclusion and future work.

2 Motivation

The REST architecture style is one of the most successful architectures designed for Web applications with the requirements including low entry barrier, extensibility, distributed hypermedia, anarchic scalability and independent deployment [11]. Many features of the REST architecture style can also benefit the IoT system requirements like low entry barrier, decentralization, scalability, robustness and easy deployment. For example, Guinard proposed the resource oriented architecture for the web of things based on REST principles [13]. Furthermore, many existing tools and techniques have supported the REST architecture style, therefore it is easy to integrate with current web technologies. As a result, it has been widely used in many IoT systems such as *Intelligent Buildings* [9], *Smart Homes* [15], *Smart Grids* [17] and *Smart Cities* [20]. Among the 39 available IoT platforms that are surveyed in [18], only 7 platforms do not have REST API.

However, using REST architecture style in the IoT systems may cause two problems in the system states verification and physical behaviours implementation. Below we will use two examples to explain them respectively.

2.1 Issue of System States Verification

To address the issue of system state verification, a scenario in the *Smart Home* is used to explain how the REST style may cause a wrong system states verification.

The scenario is to turn on/off a lamp in a room. Assume there is a controller for a lamp in the room, and it has two operations *switchOn* and *switchOff*. The typical model and design with RESTful interface for this scenario could be as Fig. 1. Based on the HTTP standards, if the response status code is “200 OK”, the operation successes, and if the response status code is “5xx”, the operation failed with service side error.

However, the problem is that even if the response of the status code “200 OK” is obtained, the whole operation cannot guarantee to be successful. The returned “200 OK” only means the controller has been successfully triggered, but the lamp may still be off for some unknown reasons, for example, due to the network problem, so the returned status code cannot reflect the real situation.

This kind of problems can be fixed by other fault tolerance mechanisms in middlewares, however, it makes the solution more complex with extra requirements on techniques and tools. Especially, some methodologies may break the constraints in the REST style, and make it more difficult to model the system states and behaviours.

This paper intends to provide a paradigm with feedback mechanism for better system states verification in the IoT systems, so the services developed in the IoT systems, especially in REST style, can be more accurate and reliable.

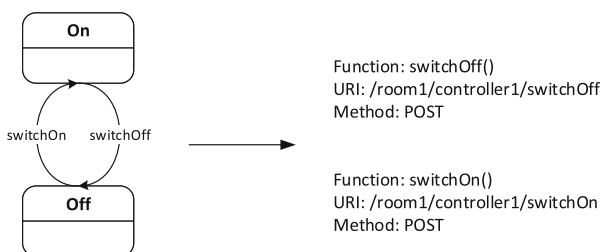


Fig. 1. Model and design with RESTful interface to turn on/off a lamp

2.2 Issue of Physical Behaviours Implementation

The second issue of physical behaviours implementation is a big problem for developing REST style services in the IoT systems. More specifically, any implementation of continuously physical behaviours with REST style services can be difficult, because the REST style services have limited operations (GET, POST, PUT, DELETE) that cannot fully match the continuously changing physical behaviours. Below we use a scenario of braking a car to explain the limitation.

Assume we need a braking service, which can brake a car based on current conditions and decisions. This scenario cannot be modelled by a simple state machine. The dynamic physical behaviours of the car can be expressed as follow:

$$\dot{s} = \frac{ds}{dt} = v, \dot{v} = \frac{dv}{dt} = a \quad (1)$$

where s represents the passage within time t with the velocity v , and a is the acceleration. From the REST style services development point of view, we need the GET methods for three variables, s , v and a first, and a POST method to call the *brake service* with parameters a and v and expected passing distance s .

However, it is impossible to ignore all disturbances and uncertainties in the physical environments, so calling a simple *brake service* with parameters a and v and expected passing distance s may cause unpredictable effects, that is, the real passing distance s' is far from s . Furthermore, it is very difficult to map the physical braking device to the braking service because quantitatively describing the action is hard. To overcome the limitations of the open-loop controller, control theory introduces feedback and a closed-loop controller that can use feedback to control states or outputs of a dynamic system, and the Fig. 2 indicates the physical behaviour. However, the traditional REST style services cannot perfectly implement this model.

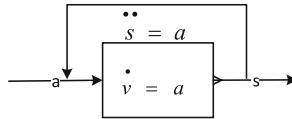


Fig. 2. The model for braking service

The paradigm proposed in this paper can also be used to enhance the functionalities of the REST to support more complex operations in physical world in a natural way, because it fits the mathematical form of calculus.

3 Feedback-Based Adaptive Service-Oriented Paradigm

In the SOA based IoT systems, we distinguish three types of different services, i.e., *Virtual Service*, *Perceptive Service* and *Actuating Service*. The three types are evaluated by the interactive patterns from the service providers to the physical environment.

Virtual Service. Most of traditional software services are virtual services with no interaction with physical environment. Even in IoT systems like smart home, most of services are still virtual services which can store temperature data or convert temperature from one unit to another, for example.

Perceptive Service. Perceptive Services are usually provided by sensors and responsible for detecting the physical environment. The perception provided from the perceptive service can be temperature, pressure or vision etc.

Actuating Service. Actuating Services are expressed as services executing real actions in the physical environment. For example, in smart home, turning on/off a light or air condition are actuating services.

An interface I of a service SP , denoted by I_s is defined by a signature and a behavioral model. In the IoT, for any given *Actuating Service*, its interface I_{ac} can be specified by *context*, *signature* and *behavioral model*. *Context* defines information depending on service requesters and service environment. *Signature* corresponds to operation profiles provided by the actuating service. *Behavioral model* is represented by Petri nets to describe the adaptive pattern.

Definition 1 (Context). We define the context as a typed relation [24], a set of ordered pairs of (d, x) where d is a dimension, T_d is the type of d and $x : T_d$. Let D denote the set of all possible dimensions, and $T = T_d | d \in D$ be the set of types associated with the dimensions. A context c is a finite relation $\{f(d, x) | d \in D \wedge x : T_d\}$. The degree of the context c is $|dom c|$.

Definition 2 (Signature). A Signature is a set of operation profiles. An operation profile is the description of an operation containing the name of an operation, with its argument types and its return type. For the actuating service interface I_{ac} , its signature is defined by a tuple $< O^{as}, O^{ps}, \Gamma >$, where O^{as} is a set of operation profiles provided by the actuating service and O^{ps} is a set of dependent operation profiles provided by other perceptive services. Γ is the function $\Gamma : O^{as} \rightarrow O^{ps}$. For any single operation profile $o_{as} \in O^{as}$, it has a set of callable operations from other perceptive services $O^{ps'} \subseteq O^{ps}$ and $O^{ps'} \neq \emptyset$, which is defined as $\gamma \in \Gamma : o_{as} \rightarrow O^{ps'}, O^{ps'} \neq \emptyset$.

Definition 3 (Behavioral Model). The behaviour in the service can be modelled as a Petri net $SN = < P, T, F, i, o >$, where P and T are disjoint sets of places and transitions. Places represent states that contain tokens with multiple attributes, and transitions represent activities that can be guarded; transitions are fired when all the tokens in the corresponding input places arrive. Places and transitions are connected through arcs.

Definition 4 (Service Interface). A service interface is a tuple $< CP, S, B >$, where: CP is a context profile, and S is a signature with its corresponding behaviour model B .

The behavioural model of FASOP is represented as a Petri net to indicate the atomic operation in *Actuating Services*. As shown in the left side of Fig. 3, before applying the paradigm, t_1 is a transition provided by the *Actuating Service* and p_1, p_2 are pre-condition and post-conditions of t_1 respectively. From the process point of view, if the operation in t_1 is a function call $< result : func(params...) >$, then p_1 is to map the function name $func$ and

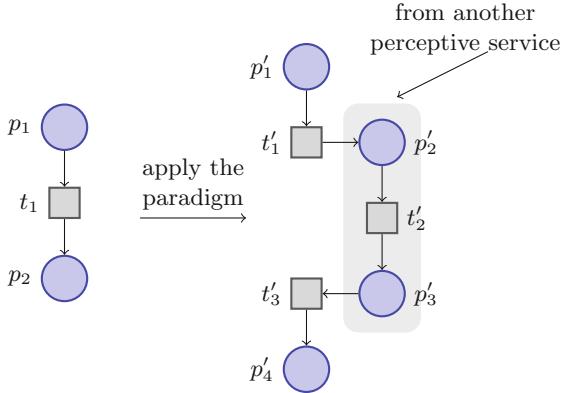


Fig. 3. The Petri Net behavioural model for an actuating service

parameters (*params...*), and p_2 is to check the return value *result*. However, if the $\langle result : func(params...) \rangle$ has any action in physical environment, it is nearly impossible to guarantee all post-conditions from the programming language level, because the post-conditions of t_1 may contain some physical effects that cannot be detected by the *Actuating Service* itself.

In our approach, the proposed paradigm is a mechanism with feedback mechanism to solve this problem at the programming language level. Feedback control is a central element of control theory, and the importance in self-adaptive systems has already been discussed in [2].

Among the three types of the services in IoT, the feedback loop can be constructed by *Actuating Services* and *Perceptive Services*. A service signature explicitly exposed by a *Actuating Service* is a set of operations that need to declare reachable *Perceptive Services* with specific operations. For a single operation profile, it is expressed as shown in Table 1. Then, any service call to *funcAS* has to pass all required parameters including context information and at least one extra service call as an available *Perceptive Service*. The behavioural model is at the right side of the Fig. 3. The service call at t_1 is changed from $\langle result : func(params...) \rangle$ to $\langle result : func(params..., PS.funcPS, t) \rangle$, which contains another function *funcPS* from another *Perceptive Service* and the latency time t which is the waiting time to get the feedback perceptions. In this way, the verification for post-conditions is more reasonable, since the post-conditions with physical properties can be verified through the perceptions of the physical environment by the *Perceptive Services*. With this new paradigm, the place p_2' can verify whether the operation *func* is operated successfully and place p_4' can eventually check if the operation *func* has desired behaviours based on the perceptions. In Fig. 4, a sequence diagram shows the detailed processes from the implementation perspective.

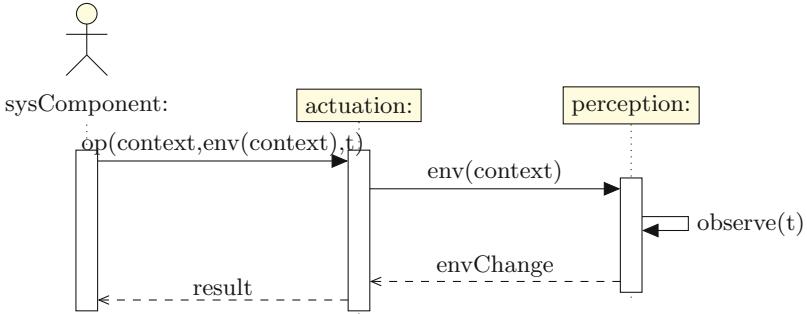


Fig. 4. Sequence diagram for calling an actuating service

Table 1. A Single operation format in actuating service

Operation name	<i>funcAS</i>
Parameters	p_1, p_2, \dots, p_n
Available operations	$PS_1.op_1, PS_2.op_1, PS_2.op_2, \dots, PS_i.op_j$

4 Extending REST for the IoT Based on FASOP

In [10], an example of using REST-based architecture server to control a robot is presented. The author concluded that REST sometimes is inconvenient compared to other RPC style web services because it does not have any functionalities like callbacks to support complex modelling with the states. The key problem is that keeping all required information in a single request to model physical behaviour while keeping stateless interactions is difficult. In physical environment, most of the continuously physical behaviours are modelled based on differential equations so it needs at least two states to express a continuous physical behaviour. Therefore at least two states in the response are needed to model the physical behaviours in any single request. With the FASOP, any single service call to an *Actuating Service* actually becomes a transaction, so all required information can be wrapped up to model physical behaviours within one request. This paradigm can be simply converted to an extra constraint for the REST and the new constraint is expressed as follow:

- Any operation from the *Actuating Services* has to operate in a complete feedback loop containing the perception to physical environment and the response need to have at least two states of the requested entity.

The high-level model we use for IoT systems is based on [16], which is defined as a tuple $RS = \langle R, I, B, \eta, C, D, \sim, OPS, RETS \rangle$, where R is a set of resources; I is a set of resource identifiers; $B \subseteq I$ is a finite set of root identifiers; $\eta : I \rightarrow R$ is a naming function, mapping identifiers to resources. C is a set of client identifiers and D is a set of data values, with an equivalence relation $\sim \subseteq (D \times D)$; OPS is

a finite set of methods; and $RETS$ is a finite set of return codes. The detailed model is similar as the model provided by [21], where the only difference is for modelling services for actuators.

Resource identifiers are modelled as URIs, represented as the following scheme:

$$URI = \text{scheme} : [/host[: port]][/]path[?query][\#fragment]$$

The descriptions of a service can be obtained by sending a GET to a particular resource via URI . Most of the service calls are at protocol level via message delivery and the main difference is on the *Actuating Service* calling. The request to an *Actuating Service* needs to contain at least one available *Perspective Service* operation.

5 Implementation Methods

In this section, we will use Java web service to express a simple implementation of this paradigm.

Based on the former definition in Sect. 3, the services in the IoT systems are concluded as: *Virtual Service*, *Perceptive Service* and *Actuating Service*. Since *Virtual Service* is just normal web service, we develop two extra interfaces: *PerceptiveService* and *ActuatingService*. To hide many network details, we assume all the services can remotely call another service from a different device based on RPC framework or Actor model [1]. Figure 5 indicates a basic example of these two interfaces.

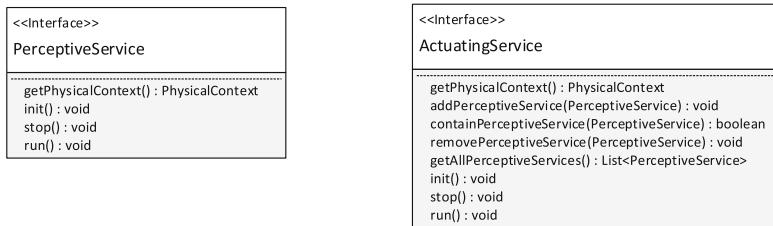


Fig. 5. The basic class diagram of the two physical interfaces

The main purpose of the interface design is to do type checking in the development. By using annotation in Java, we can restrict the developer to include at least one *PerceptiveService* as a parameter in any *Actuating Service* annotated by @WithFeedback. However, the type checking at this level needs to remotely call a method, if you want to use the JAX_RS (Java API for RESTful Web Services) [14] standard to develop REST style services, the parameters are all String type for the services mapped from the URI, thus you cannot do type checking

to confirm the *PerceptiveService* as a parameter. In this case, you need to check the *PerceptiveService* in the function of the service.

The implementation is only a lightweight version of the FASOP implementation, because we need to extend the HTTP or CoAP (Constrained Application Protocol) to fully support the FASOP, which is considered as a part of future work.

6 Case Studies

In this section, we use the two examples in the motivation section to illustrate the advantages of the FASOP.

6.1 Turn On/Off a Lamp in the Smart Home

For the scenario in the Smart Home to turn on/off a lamp in a room, the issue is that the response status code cannot express the correct system status. To solve this problem, we use the FASOP to modify the original approach and the changes are as follow:

Function: switchOn() → switchOn(PerspectiveService,t)

URI: /room1/controller1/switchOn

→ /room1/controller1/switchOn/?perceptiveservice=lightsensor&time=1 s

Method: POST

The implementation details are expressed in Fig. 6. In this implementation, the successful status code correctly reflects a guaranteed successful confirmation.

```
public class LampService implement ActuatingService{
    .....
    @POST
    @Path("/room1/controller1/switchOn/{perceptiveservice}\{time\}")
    public String switchOn(@PathParam("perceptiveservice") String p,@PathParam("time") String time){
        if(!PerceptiveServices.containskey(p)){
            switchOn();
            return "No PerceptiveService Found, Switched On";
        } else {
            switchOn();
            Thread.sleep(Integer.parseInt(time)*1000);
            String state = PerceptiveServices.get(p).getResponse();
            if(state.equals("On")){
                return "Switched On, Successfully";
            } else{ return "Failed"; } }
        }
    .....
}
```

Fig. 6. Implementation sample of using FASOP to turn on a lamp

6.2 Brake an AutoDriving Car

Compared to the traditional REST style development, the FASOP can help to transfer the physical behaviour model to software development in a more natural way. Below we use the example introduced in Sect. 2 to explain how the FASOP can help to transfer physical behaviour model to software development.

Based on the Eq. 1, in a very short time $\Delta t = t' - t$, we have the following form of the equation:

$$\dot{s} = \frac{ds}{dt} = \frac{s' - s}{t' - t} = v, \dot{v} = \frac{dv}{dt} = \frac{v' - v}{t' - t} = a = \frac{s' - s}{(t' - t)^2}$$

With the traditional REST style service development, it is very difficult to quantitatively evaluate and analyze acceleration. However, the FASOP can fit the closed-feedback model, thus the exact acceleration value can be easily evaluated via the distance and time. Furthermore, we can continuously change the acceleration via braking physically and all effect can be evaluated though the service in real-time. The function of this braking service can be as follow:

Function: braking(PerspectiveService,t)

URI: /car/brake/braking/?perceptiveservice=distancesensor&time=1 ms

Method: POST

In any moment, with this braking service, we can also predict the future passing distance s_f during the time period t_f . The predication can be based on: $s = v * t - \frac{a*t^2}{2}$ if the acceleration keeps the same.

7 Related Works

There are many researches on the development of the IoT systems to support context-awareness and adaptation. In [23], a platform is developed as ContextServ to simplify the development of context-aware Web services adopting high-level modelling language. In [3], a design for adaptation approach is proposed to support the development, deployment and execution of systems in dynamic environments by exploiting service refinement and re-configuration techniques. In [22], the MAPE-K feedback loop is used to support a synchronization and adaptation mechanism for real world process as a process-based framework. It uses a different perspective from combining processes' virtual world and real world effects to build self-adaptive IoT systems. The work can achieve a high level of autonomy and resilience against failures for physical world process. In [8], the authors provide the methodology of using model-based service oriented architecture with service composition to support self-adaptation. The work is solid and also provides fault tolerance mechanism. In [4], the service adaptation is achieved using service composition for automatic reconfiguration based on the rich interface specifications. Following this idea, they used the Discrete Time Markov Chains in a language to describe the impact of adaptation tactics and the assumption about the environment.

Our approach is a paradigm that can be used in the current service based technologies, especially for a widely used style of REST. Because of the special constraints of the REST style, the REST style services are not very suitable for context-adaptation in the IoT system. Compared to others' work, we used a different perspective of designing the Feedback-based Adaptive Service-Oriented Paradigm to support context-adaptation in service development, especially for

REST style service development which rarely supported the context-adaptation. Furthermore, we proved that this paradigm can overcome two issues in using REST style services in the IoT systems.

8 Conclusion and Future Work

The two issued caused by using the REST style services in the IoT systems are from the lack of developing the complex behaviour models in the REST style. To overcome the problem, we proposed the Feedback-based Adaptive Service-Oriented Paradigm to provide the context-adaptation ability at low level for service development, therefore the REST style services can implement complex behaviour processes based on the context-adaptation.

The implementation in this paper is a simplified version to use the current web technologies. To fully implement the FASOP in the REST style, we plan to develop a protocol by extending the HTTP or CoAP (Constrained Application Protocol) based on REST model. In addition, in the future we will also develop and deploy this paradigm in some real IoT systems.

References

1. Agha, G.A.: Actors: A model of concurrent computation in distributed systems. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab (1985)
2. Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., Shaw, M.: Engineering self-adaptive systems through feedback loops. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 48–70. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02161-9_3
3. Buccharone, A., De Sanctis, M., Marconi, A., Pistore, M., Traverso, P.: Design for adaptation of distributed service-based systems. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 383–393. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48616-0_27
4. Camara, J., Canal, C., Salaün, G.: Behavioural self-adaptation of services in ubiquitous computing environments. SEAMS **9**, 28–37 (2009)
5. Cámarra, J., Lopes, A., Garlan, D., Schmerl, B.: Adaptation impact and environment models for architecture-based self-adaptive systems. Sci. Comput. Program. **127**, 50–75 (2016)
6. Caporuscio, M., Raverdy, P.G., Issarny, V.: ubiSOAP: a service-oriented middleware for ubiquitous networking. IEEE Trans. Serv. Comput. **5**(1), 86–98 (2012)
7. Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., Chakraborty, D.: Intelligent agents meet the semantic web in smart spaces. IEEE Internet Comput. **8**(6), 69–79 (2004)
8. Cubo, J., Canal, C., Pimentel, E.: Model-based dependable composition of self-adaptive systems. Informatica **35**, 51–62 (2011)
9. Dinh, N.T., Kim, Y.: Restful architecture of wireless sensor network for building management system. KSII Trans. Internet Inf. Syst. (TIIS) **6**(1), 46–63 (2012)

10. Esteller-Curto, R., Cervera, E., del Pobil, A.P., Marin, R.: Proposal of a rest-based architecture server to control a robot. In: Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 708–710, July 2012
11. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)
12. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
13. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. Internet of Things (IOT) **2010**, 1–8 (2010)
14. Hadley, M., Sandoz, P.: Jax-rs: JavaTM api for restful web services. Java Specification Request (JSR) 311 (2009)
15. Kim, S., Hong, J.Y., Kim, S., Kim, S.H., Kim, J.H., Chun, J.: Restful design and implementation of smart appliances for smart home. In: Ubiquitous Intelligence and Computing, IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom), pp. 717–722. IEEE (2014)
16. Klein, U., Namjoshi, K.S.: Formalization and automated verification of RESTful behavior. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 541–556. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_43
17. Meloni, A., Atzori, L.: A cloud-based and restful internet of things platform to foster smart grid technologies integration and re-usability. In: IEEE International Conference on Communications Workshops (ICC), pp. 387–392. IEEE (2016)
18. Mineraud, J., Mazhelis, O., Su, X., Tarkoma, S.: A gap analysis of internet-of-things platforms. Comput. Commun. **89**, 5–16 (2016)
19. Mukhopadhyay, S.C. (ed.): Internet of Things. SSMI, vol. 9. Springer, Cham (2014). <https://doi.org/10.1007/978-3-319-04223-7>
20. Paganelli, F., Turchi, S., Giuli, D.: A web of things framework for restful applications and its experimentation in a smart city. IEEE Syst. J. **10**(4), 1412–1423 (2016)
21. Prehofer, C.: Models at rest or modeling restful interfaces for the internet of things. In: IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 251–255. IEEE (2015)
22. Seiger, R., Huber, S., Heisig, P., Assmann, U.: Enabling self-adaptive workflows for cyber-physical systems. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) BPMDS/EMMSAD -2016. LNBIP, vol. 248, pp. 3–17. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_1
23. Sheng, Q.Z., Pohlenz, S., Yu, J., Wong, H.S., Ngu, A.H.H., Maamar, Z.: Contextserv: A platform for rapid and flexible development of context-aware web services. In: Proceedings of the 31st International Conference on Software Engineering, pp. 619–622. ICSE'09, IEEE Computer Society, Washington, DC (2009). <https://doi.org/10.1109/ICSE.2009.5070570>
24. Wan, K.: A brief history of context. Int. J. Comput. Sci. Issues **6**(2), 33–42 (2009)
25. Yanwei, S., Guangzhou, Z., Haitao, P.: Research on the context model of intelligent interaction system in the internet of things. In: International Symposium on IT in Medicine and Education (ITME), vol. 2, pp. 379–382. IEEE (2011)



QoS Prediction for Reliable Service Composition in IoT

Gary White^(✉), Andrei Palade, and Siobhán Clarke

School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland
{whiteg5,paladea,siobhan.clarke}@scss.tcd.ie

Abstract. An Internet of Things (IoT) environment should respond to users' requirements by providing access to a number of component services, which can be used to create applications. To meet quality of service (QoS) requirements, these applications must be able to automatically adapt to changes in the QoS of their component services. In this paper we show how matrix factorisation (MF) based collaborative filtering can be used in a self-managing, goal-driven service model for task execution in the IoT. QoS prediction enables the goal-driven model to make adaptation decisions, allowing execution paths to dynamically adapt. This reduces failures and lessens re-execution effort for failure recovery. Results based on a QoS dataset show the suitability of the proposed mechanism in IoT, where providers are mobile and QoS values of services can change.

1 Introduction

The development of Internet of Things (IoT) technology has made it possible to connect various smart objects together through a number of communication protocols. The number of connected devices is predicted to grow to between 26 and 50 billion connected devices by 2020 [1, 2]. These devices will lead to a wide variety of services from multiple sources such as home applications, surveillance cameras, monitoring sensors and actuators [3]. These services could potentially be used in applications in many different domains including smart cities, home automation, industrial automation, medical aids and many others [4].

IoT applications make use of these services typically through a Service Oriented Architecture (SOA), where the services from the devices are discoverable, accessible and reusable in an IoT environment [5, 6]. An application can be created by a combination of multiple services and flexible service composition, which is useful to tackle potentially complex requests. There are a number of challenges associated with the environment's openness and dynamism, such as flexible service composition for services maintained by different hosts and adaptable composites to create a new service composition when the network topology changes the QoS of services in the environment. To address these challenges, existing works in service composition have used goal-driven reasoning mechanisms to achieve the user's complex goal, and classic AI backward-chaining to map the request to services available in the environment [7, 8].

Apart from the functional requirements, a key requirement for service composition and adaptation in the IoT is to choose the correct set of services, which can also satisfy the non-functional requirements [9]. Traditional composition approaches assume that the QoS values are already known, however in reality user side QoS may vary significantly due to unpredictable communication links, mobile service providers moving out of range and the heterogeneous provider environment [10]. Collaborative filtering uses the QoS values from other users in the environment to make predictions for candidate services [11, 12]. This allows for more accurate selection of the best service based on the non-functional requirements such as response time and throughput [13]. This can happen either before, or during the execution, if the QoS of the currently executing service begins to degrade. This is especially important for safety critical services such as those in healthcare, which have strict QoS requirements and where a service failing can have a serious impact.

In this paper we propose an approach for the self-management of dependable systems using collaborative filtering and goal oriented service composition [7], and present our initial results on an established QoS dataset. The remainder of the paper is organised as follows: Sect. 2 describes the background and related work, Sect. 3 presents the QoS-driven service composition and execution. Section 4 describes the experimental setup and Sect. 5 presents the results of the experiments. Section 6 outlines the conclusions and future work based on the results of the paper.

2 Background and Related Work

2.1 Decentralised Service Execution

Service composition is used to create complex applications based on available services provided by the devices in the environment. Dynamic service composition becomes increasingly difficult in IoT where there are a large number of available service providers, which rely on battery-powered, resource-constrained and potentially mobile devices to provide their services [14]. These devices can dynamically change their state due to poor wireless links, awake/sleep duty cycles or battery shortage [3]. In such an environment, a centralised mechanism is a single point of failure, which affects the reliability of the composition [15]. Recent works have used decentralised composition models to distribute the reconfiguration decisions across composition participants at runtime to improve the resilience and performance of the composition [7].

In previous work, we define a service composition mechanism that uses a goal-driven reasoning approach to model the capabilities of service providers in the environment and dynamically bind their offered services to an abstract composition request [8]. Apart from functional requirements and resilient execution, a service composition mechanism needs to satisfy the users' non-functional requirements. This requires QoS management to select the best set of concrete services in a composition and to replace these services if the QoS requirements of the application are not satisfied [16]. Most QoS-aware service composition

approaches assume that the QoS values of service candidates are already known and usually are provided directly by service providers or through third-party registries (e.g., UDDI) [17]. A service provider cannot give user-specific QoS as it can vary based on the user location and time of invocation [10]. Users can store their user-side QoS in the service registry and use the QoS values from other users in the environment to make predictions for unknown QoS values by collaborative filtering.

2.2 QoS Prediction

Traditional QoS prediction approaches in IoT have focused on the QoS prediction of currently executing services through time-series analysis [18–20]. These approaches rely on the user executing the service to generate the values, which can be used for time-series prediction. However, they make no estimates for the QoS values of candidate services, which could be switched to at runtime. This is a problem in IoT as due to the large number of candidate services it would be too time consuming to invoke even a subset of the functionally equivalent services during a runtime service composition.

An alternative approach to predicting QoS, inspired from recommender systems, is to use QoS information of similar users to make predictions about the QoS from possible services, by collaborative filtering [11, 21]. These approaches use matrix factorisation to allow the user to receive QoS predictions of services that they have not invoked, based on QoS values from similar users. Using the QoS from other similar users gives more information about candidate services, which can be chosen either at design time or during runtime service execution. This also has the additional benefit that we can gather the QoS information without harming the performance of the infrastructure with needless invocations of services to retrieve QoS values.

Some existing service composition approaches use collaborative QoS prediction in a cloud environment and only consider web services [21, 22]. Other mechanisms have used QoS prediction at design time to estimate changes in the QoS values at runtime [23]. We propose that these approaches can be used at the edge of the network in a heterogeneous IoT environment with services from a number of different devices.

3 QoS-Driven Service Composition and Execution

Figure 1 shows our middleware architecture, which is introduced in this section with focus on the Service Composition & Execution Engine and the QoS Monitor [8]. The main components are the Request Handler (RH), the Service Registration Engine (SRE), the Service Discovery Engine (SDE), the QoS Monitor and the Service Composition & Execution Engine (SCEE). The RH establishes a request/response communication channel with the user and forwards the request to the other middleware components. The SRE registers the available services in the environment. The SDE uses the backward-planning algorithm to identify the

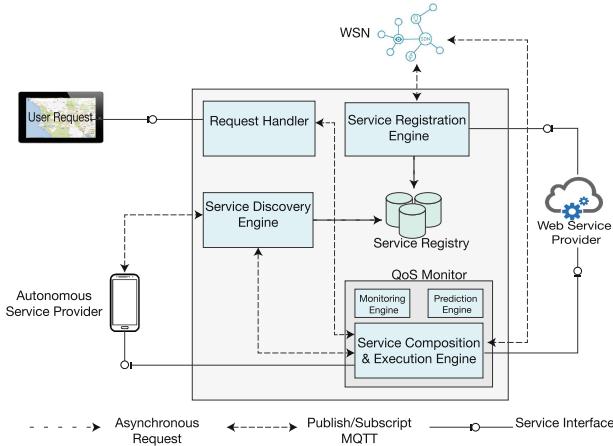


Fig. 1. Middleware architecture

concrete services, which can be used to satisfy the request and sends this list of services to the SCEE. The QoS monitor is used to monitor these services and can predict possible candidate services to switch to if one of the services begins to degrade, using the prediction engine. The SCEE will use these services to create a response for the request. This process as well as how it manages the execution in a dynamic IoT environment is explained in the following section.

3.1 Service Composition and Execution

The SCEE is responsible for the composition and execution of services discovered by the SDE. Figure 2b illustrates a list of available services in the environment identified to satisfy a user request, which was received from the RH. These services are retrieved by the RH from the environment in Fig. 2a, with services provided from different service types including web services (WS), wireless sensor networks (WSN), and autonomous service providers (ASP), who are independent mobile users with intermittent availability. The SCEE creates a list of service flows based on the concrete service providers received from the SDE. The flows are then merged based on the service description. If two or more services in the flow have the same input, the SCEE creates a guidepost to enable the invocation of one of these services based on QoS requirements (see Fig. 2b).

An execution guidepost $G = \{R_{id}, D\}$ is a split-choice control element of the composition process and maintains a set of execution directions D for a composition request R_{id} . These execution directions will be referred to as branches. Each element in the set D is defined $d_j = \{id, w, q\}$ where $j \leq |D|$. The set w represents the services in the branch and id represents the identifier of the branch. The value q reflects the branch's aggregated QoS values [7], which can be calculated according to predefined formulas [24]. The branch that maximises/minimises an objective function will be selected by the guidepost during execution.

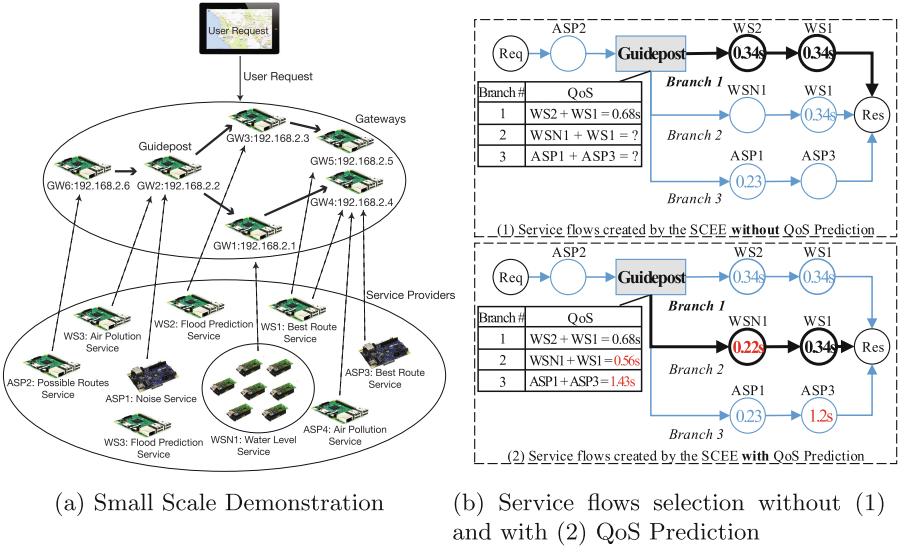


Fig. 2. Sample QoS data

In this work, we consider the response time and throughput of each branch. The formula in Eq. 1 calculates the response time by aggregating the response time value of each component service in a sequential flow [24]. In this formula, rt_i is the response time of service i . The throughput value is calculated using the formula in Eq. 2, which selects the lowest throughput $\min(th_i)$ offered by the services in a sequential flow [24]. These formulas require the response time and throughput values of each service component in the flow to calculate the aggregate values. It is possible that these values could not be calculated if the required QoS data is missing or is out-of-date.

$$\text{Response Time (RT)} = \sum_{i=1}^n rt_i \quad (1)$$

$$\text{Throughput (T)} = \min(th_i) \quad (2)$$

To address this problem, the QoS monitor uses QoS prediction to predict QoS values across each branch stored in the guidepost. Figure 2b shows the flows created by the SCEE for User 4 (U_4 in Fig. 3). The response time values recorded during service discovery phase were 0.34s for service provider WS_2 , 0.34s for WS_1 and 0.23s for ASP_1 . The response time values for WSN_1 and ASP_3 were not recorded. When the execution reaches Guidepost G, we can only aggregate the response time values for Branch 1 (Part 1 in Fig. 2b), which is not optimal. If the composition selects the branch with the lowest reported QoS values, it will select Branch 3, which is also not optimal. Only when we use the predicted values for the missing service QoS does the composition choose the optimal Branch 2 (Part 2 in Fig. 2b).

3.2 Collaborative QoS Prediction

Accurate QoS prediction of candidate services is a fundamental component of goal driven service composition. Incorrect predictions may cause compositions and adaptations to have worse QoS, which could lead to SLA violations. On-line prediction approaches have been proposed to detect service failure and QoS deviations of the currently executing services [25, 26]. Other approaches propose to collecting QoS values by invoking the candidate services, but this is not scalable in IoT due to the large amount of candidate services [27].

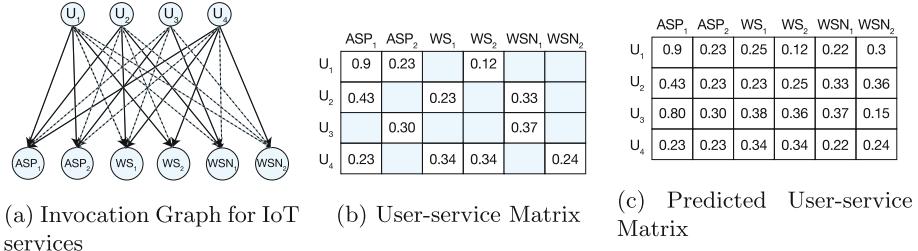


Fig. 3. Demonstration of QoS prediction in IoT

To provide QoS values on m IoT services for n users, one needs to invoke at least $n \times m$ services, this is almost impossible in an IoT environment where we expect a large number of services and users. Without this QoS information, the service execution engine cannot select the optimal components based on their QoS and must make a choice based on whatever QoS values are available. This leads to choosing potentially non-optimal services, which can cause service degradation at runtime and service execution errors. Figure 4 shows that the same service can have different values even for the same quality factors such as response time and throughput for different users and comes from a real life dataset [10].

We use collaborative filtering, which identifies users who share similar characteristics (e.g., location, response time, etc.) to make predictions of what QoS they will receive when executing a service. The QoS value of IoT service s observed by user u can be predicted by exploring the QoS experiences from a user similar to u . A user is similar to u if they share similar characteristics, which can be extracted from their QoS experiences on different services by collaborative filtering. By sharing local QoS experience among users these approaches can predict the QoS value of a range of IoT services including ASP, web services and WSN even if the user u has never invoked the service s before [28].

We give a demonstration based on a subset of the implementation in Fig. 2a, where we have a number of different service providers, who are able to provide functionally equivalent services from heterogeneous devices. We model this as a bipartite graph $G = (U \cup S, E)$, such that each edge in E connects a vertex in U to S . Let $U = \{u_1, u_2, \dots, u_4\}$ be the set of component users and

$S = \{ASP_1, ASP_2, \dots, WSN_2\}$ denote the set of IoT services and E (solid lines) represent the set of invocations between U and S . Given a pair (i, j) , $u_i \in U$ and $c_j \in S$, edge e_{ij} corresponds to the QoS value of that invocation. Given the set E the task is to predict the weight of potential invocations (broken lines).

We visualise the process of matrix factorisation for the demonstration in Fig. 3b, in which each table entry shows an observed weight in Fig. 3a. By using the latent factor model [29] a number of algorithms first factorise the sparse user-component matrix and then use $V^T H$ to approximate the original matrix, where the low-dimensional matrix V denotes the user latent feature space and the low-dimensional matrix H represents the low-dimensional item latent feature space. The latent feature space represents the underlying structure in the data, computed from the observed features using matrix factorisation. As the matrices V and H are dense it is then possible to use a neighbourhood-based collaborative method, as shown in Fig. 3c.

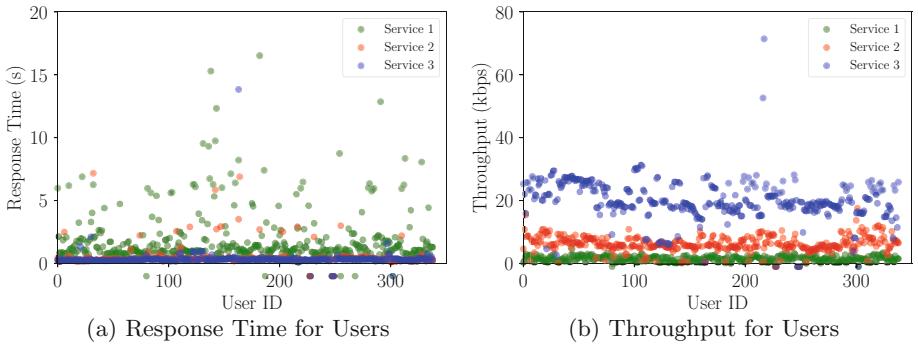


Fig. 4. Sample QoS data

4 Experimental Setup

4.1 Dataset Description

Invoking thousands of IoT services in the wild is difficult because some of the services may have limited range and may not be publicly available on the Internet. To evaluate the prediction quality of these approaches, we use a QoS dataset, which consists of a matrix of response time and throughput values for 339 users by 5,825 services [10], which would be provided by the monitoring component.

4.2 Metrics

The predictive composition process uses the predicted values generated through collaborative filtering to choose the optimal flow. The composition process without the predicted values has access to the percentage of QoS values given by the

matrix density. Once the optimal flow has been selected we report the actual values based on the original data. Two metrics are considered in this work: response time and throughput. The response time for each branch is aggregated using Eq. 1. The branch which has the lowest response time is used in the composition. The throughput value for each branch is aggregated using Eq. 2. The branch which has the highest throughput is used in the composition.

4.3 Performance Comparison

We compare the performance of the flow generated using the predicted values from the CloudPred collaborative filtering algorithm to the flow generated using no predicted values [11]. We show the optimal service composition to compare how each of the compositions approach the optimal solution. To evaluate this, we divided the available set of 5,825 services for each user into sub-sets of services each of size 50. We used each sub-set to create branches that can be used in a guidepost. The number of services in a branch was set to 20. The value of each branch was calculated by aggregating the QoS value assigned to each service in the branch. These values were stored in the guidepost and used for branch selection during composition execution. The branch selection uses a function, which minimises the response time and maximises the throughput. This allows the branch with the smallest response time to be selected and follow the execution of the guidepost. We conducted the experiment for a number of different matrix densities from 10% to 90% to show how the results of both approaches change as they get access to more data about the users. The results are shown for 1 user and an aggregation of 100 different users.

5 Results

Figure 5a and b illustrate the response time of service composition, with and without prediction, for 1 and 100 users. In Fig. 5a we see the response time results for a number of different matrix densities for one user. In this case the prediction composition shows a large improvement compared to the composition that has no prediction. As the matrix density increases we can see that the composition without prediction gets closer to the optimal value as it has access to more QoS values. Figure 5b shows the response times for the service flows averaged over 100 users. This graph follows the same trend as one user with a larger difference in response time between the composition approaches.

Figure 5c and d show how QoS prediction affects the throughput of the composed services. Figure 5c shows the results for one user, when the density is less than 20% the prediction approach shows a clear improvement with a greater throughput value. However at 30% density, we see that composition without prediction actually performs better than the predictive approach. When the matrix density is greater than 40%, both approaches are able to find the optimal composition. Figure 5d shows the throughput values averaged over 100 users. In this case, we can see that the predictive approach has a much larger throughput

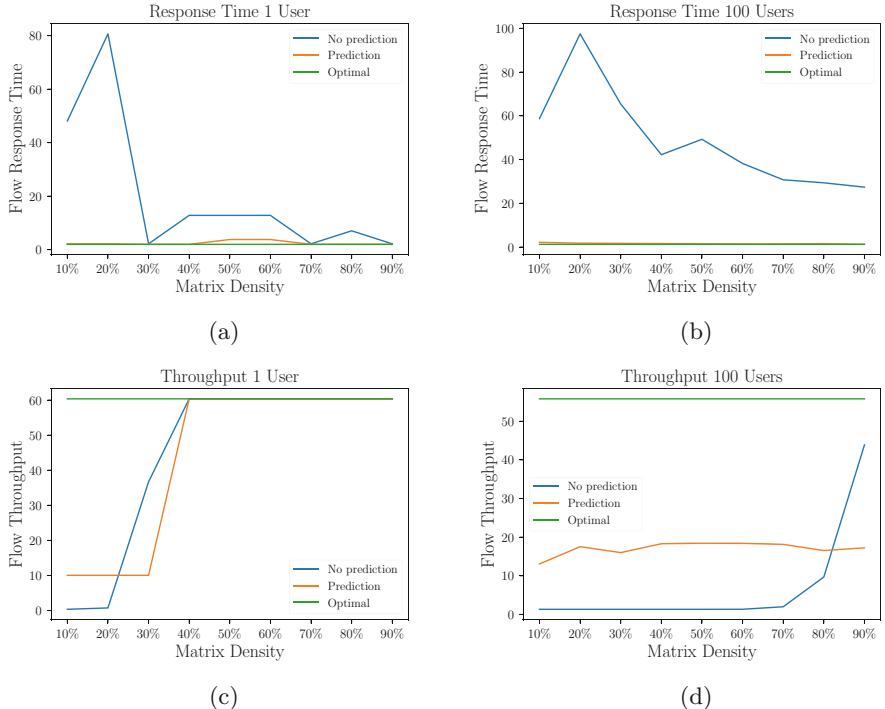


Fig. 5. Impact of prediction on service composition flows

value than the approach without prediction when the density is less than 80%. When the matrix density is greater than this, the composition without prediction improves and almost reaches the optimal throughput value.

5.1 Threats to Validity

To assess the validity of the results, we take into account areas where bias could have been introduced, following the guidelines introduced by Peterson and Gencel [30]. In particular, we consider the interpretive validity and repeatability of the experiments.

Interpretive Validity. The interpretive validity is the extent to which the conclusions from the experiments are reasonable given the data. In the results section we present the results for each of the matrix densities to allow the reader to validate the conclusions. We show the results for an individual user as well as an aggregation of 100 users to show a comprehensive evaluation of the approaches.

Repeatability. Repeatability allows for the complete repetition of the experiment to verify the results and requires detailed reporting of the research method.

In the experimental setup section, we give a detailed description of the data, metrics and the approaches that were used. In Sect. 3 we describe the goal-driven composition algorithm and collaborative filtering algorithm, which allows for the repeatability of the experiment to validate the results. The goal-driven service composition mechanism was introduced by Chen et al. [7] and an implementation of this mechanism in an IoT scenario is presented in our previous work [8]. This should enable future proposals to reproduce our results and develop new QoS prediction models for service composition in a decentralised setting based on our source code, which is available on request. In future work we will test the prediction composition in a real life environment, which although reducing the repeatability will increase the internal validity of the results.

6 Conclusion and Future Work

The results of the experiments have a number of interesting conclusions, which can be used to outline future work. The results from the response time dataset are encouraging with predictive composition showing clear improvement over all the matrix densities. For the throughput dataset, the predictive approach showed clear improvement for low matrix densities ($\leq 80\%$) when averaged over 100 users, but for densities greater than this, the composition without prediction values produced better results. There is a clear difference in the response time and throughput results, with the response time for the predictive composition approach selecting close to the optimal flow for most densities. When averaged over 100 users, the throughput results for the predictive composition do not get closer to the optimal value as the density increases. This can be attributed to the different data scales, which can introduce larger errors for the throughput predictions [28], that can result in selecting the wrong branch.

In IoT, because of the large number of services available, we would expect that only a small percentage would have relevant, up-to-date QoS information in the time period required (e.g., within 15 min). This makes the results for low matrix densities from 10%–30% particularly important, as they illustrate that we need only a small number of users in the environment to report values to achieve almost optimal results.

As future work, we will evaluate the overhead introduced by the QoS prediction mechanism when dynamically composing IoT services in a real-world environment. We also aim to investigate a number of alternative techniques for preprocessing the data and conducting similarity comparison between the users and the different types of service providers (e.g. data smoothing, clustering, PCA, etc.).

Acknowledgments. This work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885.

References

1. Bauer, H., Patel, M., Veira, J.: The internet of things: Sizing up the opportunity. McKinsey (2014)
2. Evans, D.: The internet of things: How the next evolution of the internet is changing everything. Cisco Internet Bus. Solut. Group **1**, 14 (2011). CISCO White paper
3. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutor. **17**(4), 30, Fourthquarter (2015)
4. Bellavista, P., Cardone, G., Corradi, A., Foschini, L.: Convergence of MANET and WSN in IOT urban scenarios. IEEE Sens. J. **13**(10), 3558–3567 (2013)
5. Ibrahim, N., Mouel, F.L.: A survey on service composition ware in pervasive environments. arXiv preprint [arXiv:0909.2183](https://arxiv.org/abs/0909.2183) (2009)
6. Raychoudhury, V., Cao, J., Kumar, M., Zhang, D.: Middleware for pervasive computing: a survey. Pervasive Mob. Comput. **9**(2), 177–200 (2013). Special Section: Mobile Interactions with the Real World
7. Chen, N., Cardozo, N., Clarke, S.: Goal-driven service composition in mobile and pervasive computing. IEEE Trans. Serv. Comput. **PP**(99), 1 (2016)
8. Cabrera, C., Li, F., Nallur, V., Palade, A., Razzaque, M., White, G., Clarke, S.: Implementing heterogeneous, autonomous, and resilient services in IOT: an experience report. In: 2017 IEEE 18th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE (2017)
9. Metzger, A., Chi, C.H., Engel, Y., Marconi, A.: Research challenges on online service quality prediction for proactive adaptation. In: 2012 First International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube), pp. 51–57, June 2012
10. Zheng, Z., Zhang, Y., Lyu, M.R.: Investigating qos of real-world web services. IEEE Trans. Serv. Comput. **7**(1), 32–39 (2014)
11. Zhang, Y., Zheng, Z., Lyu, M.R.: Exploring latent features for memory-based qos prediction in cloud computing. In: 2011 IEEE 30th International Symposium on Reliable Distributed Systems, pp. 1–10, October 2011
12. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Online QOS prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. **PP**(99), 1 (2017)
13. White, G., Nallur, V., Clarke, S.: Quality of service approaches in IOT: a systematic mapping. J. Syst. Softw. **132**, 186–203 (2017). <http://www.sciencedirect.com/science/article/pii/S016412121730105X>
14. Palade, A., Cabrera, C., White, G., Razzaque, M., Clarke, S.: Middleware for internet of things: a quantitative evaluation in small scale. In: 2017 IEEE 18th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). IEEE (2017)
15. Bronsted, J., Hansen, K.M., Ingstrup, M.: Service composition issues in pervasive computing. IEEE Pervasive Comput. **9**(1), 62–70 (2010)
16. Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G.: Dynamic QoS management and optimization in service-based systems. IEEE Trans. Softw. Eng. **37**(3), 387–409 (2011)
17. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service qos prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013)

18. Ye, Z., Mistry, S., Bouguettaya, A., Dong, H.: Long-term qos-aware cloud service composition using multivariate time series analysis. *IEEE Trans. Serv. Comput.* **9**(3), 382–393 (2016)
19. Amin, A., Grunske, L., Colman, A.: An automated approach to forecasting qos attributes based on linear and non-linear time series modeling. In: 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 130–139, September 2012
20. Amin, A., Colman, A., Grunske, L.: An approach to forecasting qos attributes of web services based on ARIMA and GARCH models. In: 2012 IEEE 19th International Conference on Web Services, pp. 74–81, June 2012
21. Lo, W., Yin, J., Deng, S., Li, Y., Wu, Z.: An extended matrix factorization approach for QOS prediction in service selection. In: 2012 IEEE Ninth International Conference on Services Computing, pp. 162–169, June 2012
22. Zheng, Z., Lyu, M.R.: Collaborative reliability prediction of service-oriented systems. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 35–44. ACM (2010)
23. Li, M., Huai, J., Guo, H.: An adaptive web services selection method based on the QOS prediction mechanism. In: 2009 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, WI-IAT 2009, vol. 1, pp. 395–402. IEEE (2009)
24. Ben Mabrouk, N., Beauche, S., Kuznetsova, E., Georgantas, N., Issarny, V.: QoS-aware service composition in dynamic service oriented environments. In: Bacon, J.M., Cooper, B.F. (eds.) *Middleware 2009*. LNCS, vol. 5896, pp. 123–142. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10445-9_7
25. Wang, C., Pazat, J.L.: A two-phase online prediction approach for accurate and timely adaptation decision. In: 2012 IEEE Ninth International Conference on Services Computing, pp. 218–225, June 2012
26. Geubelen, D., et al.: Qos prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset. *Inf. Sci.* **268**, 397–424 (2014). New Sensing and Processing Technologies for Hand-based Biometrics Authentication
27. Jiang, B., Chan, W.K., Zhang, Z., Tse, T.H.: Where to adapt dynamic service compositions. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, NY, USA, pp. 1123–1124. ACM, New York (2009)
28. White, G., Palade, A., Cabrera, C., Clarke, S.: Quantitative evaluation of QOS prediction in IOT. In: 3rd International Workshop on Recent Advances in the Dependability Assessment of Complex Systems, June 2017
29. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. *Nips* **1**(1), 1–2 (2007)
30. Petersen, K., Gencel, C.: Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), pp. 81–89, October 2013



Checking and Enforcing Security Through Opacity in Healthcare Applications

Rym Zrelli^{1(✉)}, Moez Yeddes², and Nejib Ben Hadj-Alouane¹

¹ OASIS Research Lab (ENIT), University of Tunis El Manar, Tunis, Tunisia
rym.zrelli@gmail.com

² OASIS Research Lab (INSAT), University of Carthage, Tunis, Tunisia

Abstract. The Internet of Things (IoT) is a paradigm that can tremendously revolutionize health care thus benefiting both hospitals, doctors and patients. In this context, protecting the IoT in health care against interference, including service attacks and malwares, is challenging. Opacity is a confidentiality property capturing a system's ability to keep a subset of its behavior hidden from passive observers. In this work, we seek to introduce an IoT-based heart attack detection system, that could be life-saving for patients without risking their need for privacy through the verification and enforcement of opacity. Our main contributions are the use of a tool to verify opacity in three of its forms, so as to detect privacy leaks in our system. Furthermore, we develop an efficient, Symbolic Observation Graph (SOG)-based algorithm for enforcing opacity.

1 Introduction

Real-world usage of IoT in health-care necessitates the dealing with new security challenges. In fact, and since this type of application would handle medical and personal information, their employment carries serious risks for personal privacy. Accordingly, it is paramount to protect any sensitive data against deduction by third-parties to avoid the compromise of privacy. The most common security preservation practice is the use of cryptographic techniques. However, these techniques do not provide perfect security as the inference of critical information from non-critical ones remains a possibility. The discovery of vulnerabilities of simple crypto-systems like that of the Needham-Schroeder public key protocol [10] proved that cryptography is not enough to guarantee the privacy of information. Furthermore, the various techniques available are computationally intensive. This is why they cannot be immediately adopted in IoT where the network nodes are powered by battery. To facilitate the adoption of IoT in health-care, we need formal (preferably automated) verification of security properties. Formal verification ensure that the system's design conforms to the desired behavior. Information flow properties are the most formal security properties. In fact, various ones have been defined in the literature including non-interference, intransitive non-interference and others (e.g. secrecy, and anonymity). Interested in

confidentiality properties, we consider opacity, a general information flow property, to analyze IoT privacy in a heart attack detection system. Opacity's main interest is to formulate the need to hide information from a passive observer. It was first introduced in [12] and was later generalized to transition systems [4]. It has since, been studied several times allowing the formal verification of system models. Its wide study led to the birth of several variants as well as verification and enforcement techniques. If classified according to the security policy, then we are dealing with simple, K -step, initial, infinite as well as strong and weak opacity alongside their extensions (e.g., K -step weak and K -step strong opacity). The efforts of these studies also made possible not only opacity verification, but also its assurance via supervision [5], [14] or enforcement [7]. Several IoT-based solutions [1,8] for healthcare are known in the literature to deal with privacy issues. A key limitation of these studies is that they have been using cryptographic methods.

In this paper, we wish to show the practical use of our SOG-based approach and the relevance of the use of opacity in the real world through the synthesis of an opaque IoT-based heart attack detection system. Building on the SOG-based verification approach developed in [3], the purpose is to verify opacity in three of its forms (simple, K -step weak opacity and K -step strong opacity) to detect security violations in our synthesized system. Then to contribute an algorithmic approach that enforces simple opacity by padding the system with minimal dummy behavior.

This paper is organized as follows: Sect. 2 establishes all necessary basic notions including the SOG structure and the opacity property. In Sect. 3, we detail the case study. In Sect. 4, we illustrate the practical usefulness of the opacity verification approach in the heart attack detection system. Section 5 details our proposed approach to enforce simple opacity. Finally, we conclude in Sect. 6, and list some potential future works.

2 Preliminaries

2.1 Petri Nets, WF-net and oWF-nets

To model the services under consideration in our case study, we use Petri nets. A service can be considered as a control structure describing its behavior in order to reach a final state. We can represent it using a Workflow net, a subclass of Petri nets. A WF-net satisfies two requirements: it has one input place i and one output place o , and every transition t or place p should be located on a path from place i to place o . To model the communication aspect of a service, we can use open Work-Flow nets which is enriched with communication places representing the (asynchronous) interface. Each communication place represents a channel to send or receive messages to or from another oWF-net.

Definition 1 (oWF-net [11])

An open Work-Flow net is defined by a tuple $\mathcal{N} = (P, T, F, W, m_0, I, O, m_f)$:

- (P, T, F, W) is a WF-net;
 - P is a finite set of places and T a finite set of transitions;
 - F is a flow relation $F \subseteq (P \times T) \cup (T \times P)$;
 - $W : F \rightarrow \mathbb{N}$ is a mapping allocating a weight to each arc.
- m_0 is the initial marking;
- I is a set of input places and O is a set of output places ($I \cup O$: the set of interface places).
- m_f is a final marking.

Having the same semantics as Petri nets, the behavior of WF-nets and oWF-nets can be represented by Labeled Transition Systems (LTS).

2.2 Labeled Transition System

An LTS is defined as follows:

Definition 2 (Labeled Transition System)

A Labeled Transition System is a 4-tuple $\mathcal{G} = (Q, q_{init}, \Sigma, \delta)$:

- Q : a finite set of states;
- q_{init} : the initial state;
- Σ : actions' alphabet;
- $\delta : Q \times \Sigma \rightarrow Q$: the transition function where: $q, q' \in Q$ and $\sigma \in \Sigma$, $\delta(q, \sigma) = q'$ meaning that an event σ can be executed at state q leading to state q' .

The language of an LTS \mathcal{G} is defined by $L(\mathcal{G}) = \{t \in \Sigma^*, q_0 \xrightarrow{t} q_f\}$. An LTS can be considered as an automaton where all states are accepting final states.

To reflect the observable behavior of an LTS, we specify a subset of events $\Sigma_o \subseteq \Sigma$ and $\Sigma - \Sigma_o = \Sigma_u$ where Σ_o is the set of events visible to a given observer and Σ_u is the set of events which are invisible to said-observer. The behavior visible is defined by the projection P_{Σ_o} from Σ^* to Σ_o^* that removes from a sequence in Σ^* all events not in Σ_o . Formally, $P_o : \Sigma^* \rightarrow \Sigma_o^*$ is defined:

$$\begin{cases} P_{\Sigma_o}(\epsilon) = \epsilon; \\ P_{\Sigma_o}(u \cdot \sigma) = \begin{cases} P_{\Sigma_o}(u) & \text{if } \sigma \notin \Sigma_o; \\ P_{\Sigma_o}(u) \cdot \sigma & \text{otherwise.} \end{cases} \end{cases} \quad \text{Where: } \sigma \in \Sigma \text{ and } u \in \Sigma^*.$$

2.3 Opacity

Opacity's main interest is in capturing the possibility of using observations and prior-knowledge of a system's structure to infer secret information. It reflects a wide range of security properties. Opacity's parameters are a secret predicate, given as a subset of sets or traces of the system's model, and an observation function. This latter captures an intruder's abilities to collect information about the system. A system is, thus, opaque w.r.t. the secret and the observation function, if and only if for every run that belongs to the secret, there exists another run with a similar projection from the observer's point of view and that does not belong to the secret [5, 6, 9]. In this paper, we focus on 3 opacity variants as defined in [6]: simple, K -step weak and K -step strong opacity.

Definition 3 (Simple opacity [6])

Given an LTS $\mathcal{G} = (Q, q_0, \Sigma, \delta)$ with $\Sigma_o \subseteq \Sigma$ is the set of observable events and $S \subseteq Q$ is the set of secret states. The secret $S \subseteq Q$ is opaque under the projection map P_{Σ_o} iff (G, P_{Σ_o}) – opaque iff: $\forall u \in L_S(G), \exists v \in L(\mathcal{G}) : (v \approx_{\Sigma_o} u) \wedge (v \notin L_S(G))$.

While simple opacity deals with the non-discloser of the fact that the system is currently in a secret state, K -step weak opacity ensures that the system wasn't in a secret state K observable events ago, and K -step strong opacity formulates the need to make sure that, K -steps backwards, the system does not end, and have not crossed any secret states.

2.4 Symbolic Observation Graph

The SOG is an abstraction of the reachability graph. It is constructed by exploring a system's observable actions which are used to label its edges. The unobservable actions are hidden within the SOG nodes named aggregates. The definition of an aggregate and that of the SOG are given in the following:

Definition 4 (Aggregate)

Given an LTS $\mathcal{G} = (Q, q_0, \Sigma, \rightarrow, \delta)$ with $\Sigma = \Sigma_o \cup \Sigma_u$. An aggregate a is a non empty set of states satisfying: $q \in a \Leftrightarrow \text{Saturate}(q) \subseteq a$ where: $\text{Saturate}(q) = \{q' \in Q : q \xrightarrow{w} q' \text{ and } w \in \Sigma_u^*\}$.

Definition 5 (Deterministic SOG)

A deterministic SOG(\mathcal{A}) associated with an LTS $\mathcal{G} = (Q, q_0, \Sigma_o \cup \Sigma_u, \delta)$ is an LTS $(A, a_0, \Sigma_o, \Delta)$ where:

1. A a finite set of aggregates with:
 - (a) $a_0 \in A$ is the initial aggregate s.t. $a_0 = \text{Saturate}(q_0)$;
 - (b) For each $a \in A$, and for each $\sigma \in \Sigma_o$, $\exists q \in a, q' \in Q : q \xrightarrow{\sigma} q' \Leftrightarrow \exists a' \in A : a' = \text{Saturate}(\{q' \in Q, \exists q \in a \text{ with } q \xrightarrow{\sigma} q'\}) \wedge (a, \sigma, a') \in \Delta$;
2. $\Delta \subseteq A \times \Sigma_o \times A$ is the transition relation.

3 Motivating Scenario

Heart disease is the first cause of morbidity and mortality in the world, accounting for 28.30% of total deaths each year in Tunisia alone [13]. Investment in preventive health care such as the use of IoT monitoring devices may help lower the cost of processing and the development of serious health problems. Integrating clinical decisions with electronic medical records could decrease medical errors, reduce undesirable variations in practice, and improve patient outcomes.

Our case study considers IoT integration with cloud computing. We use a connected bracelet, fog nodes, a private and a public Cloud, and a mobile application, which together form a medical application. This latter provides continuous monitoring of the vital data of a given patient. Regular or routine measurements could help to detect the first symptoms of heart malfunction, and makes it possible to immediately trigger an alert. The vital information collected by the

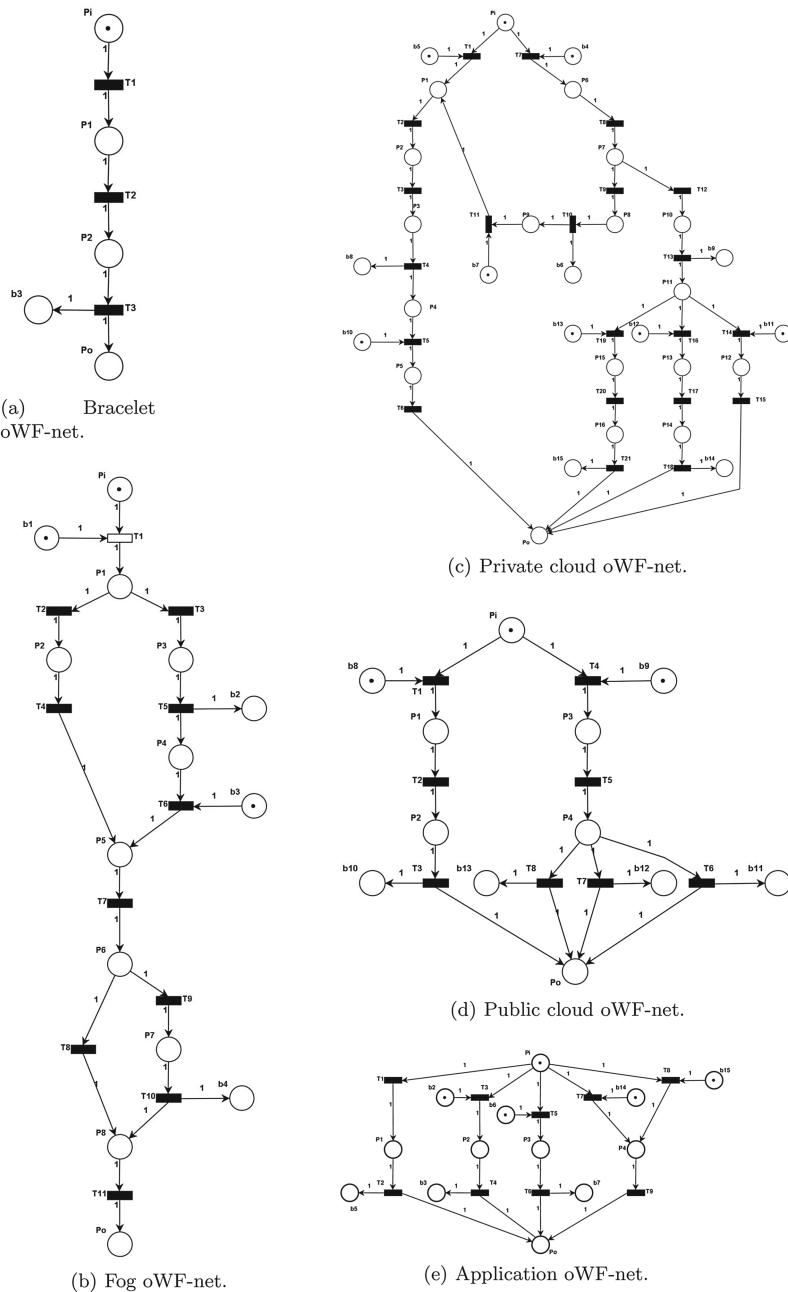
bracelet includes cardiac activity, blood pressure, oxygen levels and, temperature. As mentioned earlier, we consider an IoT application in a hybrid cloud/fog environment. The cloud [16] is considered as a highly promising approach to deliver services to users, and provide applications with low-cost elastic resources.

Public clouds provide cheap scalable resources. Making it useful for analyzing the patient's data which would be costly as it requires extensive computing and storage resources. However, we must take into account that storage of health records on a public environment is a privacy risk. To avoid such security leaks, we could deploy the application on a secure private cloud. But seeing this latter's limited resources, this may degrade the overall performance. To prevent this, the workflow can be partitioned between a private cloud and a public one. Therefore, the confidential medical data will be processed on the private cloud. Other workflow actions can be deployed on the public cloud dealing with anonymized data. The use of a cloud-based framework poses the problem of delay when sending and receiving data between the objects and geographically far cloud resources thus jeopardizing the patients' well-being given that triggering timely responses is the purpose of this data. To resolve this issue, data gathering can be moved from the cloud domain to that of the fog [2]. Bringing this action closer to the connected object shortens the transmission time, and reduces the amount of data transferred to the cloud. The proposed workflow is described as follows:

- A patient may register via the mobile app by entering his information. This information include personal data and medical history (personal and family medical histories, surgical history, drug prescriptions, and the doctors' notes).
- The patient's medical history is then transmitted to the private cloud. After reception, this latter anonymizes the data by stripping off all that could identify the patient leaving only medical data, which it sends to the public cloud.
- The public cloud receives the anonymized data, and proceeds to the classification attaching to each medical file a class.
- The patient is equipped with a measuring bracelet connected to the processing components (Fog nodes). The data sent to the fog domain is a set of vital data recorded over a period of time.
- The fog node collects the data then compares it to its predecessors, searching for any vital signs changes. When the node determines that a change has occurred, it sends the data to the private cloud.
- The private cloud links the gathered data with the patient, transmitting this data and the class ascribed to the patient, to the public cloud.
- The public cloud reads the data, analyzes it, and then provides results. When the risk of heart attack is detected, it immediately notifies the patient's app.

4 Modeling and Verification

The case study contains five services, namely, a connected bracelet (Br), a fog node (Fog), a private cloud (CPr), a public cloud (CPub), and a smartphone application (App). Figure 1 depicts the oWF-nets of the Br, Fog, CPr, CPub and the App, respectively. We note that the transitions entailing the sending

**Fig. 1.** Case study oWF-nets.

(respectively reception) of a messages are indicated by adding a ! (respectively a ?) mark. In this case study, we want to illustrate the ability of the SOG-based verification approach to meet privacy demands. The first step is to create the underlying LTS of each oWf-net. Secondly, we identify the observable and unobservable actions of each net as well as the secret states. Then, we build the SOG models from each net's LTS verifying, at the same time, their opacity.

The Br workflow (Fig. 1(a)) starts by collecting data (T_1), which will then be sent to the closest Fog node. Next it creates the message comprising the data (T_2) and sends this message ($T_3?$). Not having any security requirements for the bracelet, thus, there is no need to check its opacity.

The Fog WS (Fig. 1(b)) has an internal set of operations, and a set of external cooperative ones. After receiving the data ($T_1!$), we consider two scenarios. The first is when the Fog communicates for the 1st time with the bracelet (T_3). In this case, it sends a request ($T_5?$) to the App to retrieve data from the patient's medical history. Then, it will receive these data through ($T_6!$). The second scenario begins by selecting the last recorded data (T_4). The next step is to compare (T_7) the data retrieved by one of the mentioned scenarios with the data sent by the Br. When the node detects a change in values (T_9), it will immediately transmit the data to CPr ($T_{10}?$). If there is no change (T_8), the Fog doesn't perform any processing. Finally, the new data will be stored locally in the Fog (T_{11}). To ensure the privacy of fog secret information, we define the secret state $S = \{S_6\}$ which is related to receiving patient's medical history. To conform with the security needs, the observable transitions of the Fog are $\Sigma_o = \{T_1!, T_5?, T_6!, T_{10}?\}$, while the unobservable part is $\Sigma_u = \{T_2, T_3, T_4, T_7, T_8, T_9, T_{11}\}$. Using this data, we proceed to the opacity verification which is done while creating the SOG-abstraction of the model. We get the SOG in Fig. 2(a) and we can conclude that the fog's SOG is both simple, and K -step weakly and strongly opaque.

The CPr workflow (Fig. 1(c)) contains two scenarios. The first one starts by receiving the data of a registered patient ($T_1!$). The CPr subsequently proceeds with the recording (T_2) and the anonymization (T_3) of the received data. The anonymised data will then be transmitted to the CPub ($T_4?$). After receiving ($T_5!$) the class, this latter is associated with the patient (T_6). The second scenario starts when the CPr receives ($T_7!$) the data sent by the Fog. The CPr combines the data with the patient by searching for its ID (T_8). If the ID cannot be found (T_9), the CPr sends a request to the App so that the patient re-enter his information ($T_{10}?$). Thereafter, it receives the requested data ($T_{11}!$) and it pursues the first scenario. For the second case, when the ID is found, the CPr transmits the data and the class to which the patient belongs to the CPub ($T_{13}?$). Afterwards, the CPr receives and records respectively 3 types of messages, each one belongs to an alert type: low ($T_{14}!$ & T_{15}), medium ($T_{16}!$ & T_{17}) and high ($T_{19}!$ & T_{20}). To protect the privacy of patients, the CPr need to hide the update procedure performed on the patient's personal information. It must keep secret the states related to the patient registration (S_4 & S_{16}) and the anonymization of his data (S_7 & S_{21}). It is also required to withhold secret the states related to sending alerts (S_{22} & S_{23}). So the set of

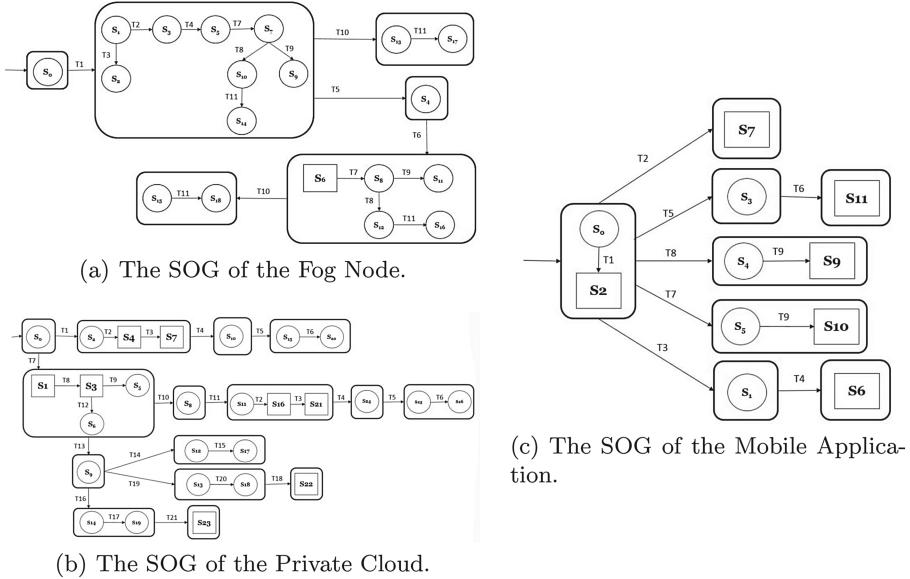


Fig. 2. The SOGs of the case study WSs.

secret states for the CPr is $S = \{S_1, S_3, S_4, S_7, S_{16}, S_{21}, S_{22}, S_{23}\}$, where S_1 stands for the marking related to the reception of the data sent by the fog, while S_3 reflects that related to patient ID search. The observable transitions of the CPr are $\Sigma_o = \{T_1!, T_4?, T_5!, T_7!, T_{10}?, T_{11}!, T_{13}?, T_{14}!, T_{16}!, T_{18}?, T_{19}!, T_{21}?\}$, while the unobservable ones are $\Sigma_u = \{T_2, T_3, T_6, T_8, T_9, T_{12}, T_{15}, T_{17}, T_{20}\}$. With this configuration, we conduct the verification and get the SOG in Fig. 2(b). Thus, the CPr workflow is not opaque and is not k-step weakly and strongly opaque. Indeed, the two secret states S_{22} and S_{23} , each belonging to an aggregate that doesn't hold other non-secret states. An attacker can then disclose secret information after the traces $T_7T_{13}T_{16}T_{18}$ and $T_7T_{13}T_{19}T_{21}$. The CPr service is therefore unsafe and needs to be improved. Taking into account that the CPub is available for public use, we don't have secrets to be hidden from an external observer. So, we will only describe the CPub actions (Fig. 1(d)) and we won't proceed the opacity verification. The first set of CPub actions concerns the internal operations which include the processing of the data sent by the CPr: the classification (T_2) and the prediction (T_5) which aims to detect the risk of heart attack. As regards the external operations, the CPub receives two messages from the CPr. The first one (T_1) includes the anonymised data and the second (T_4) includes the data collected by the Br and the class to which the patient belongs. In response to the received messages, the CPub sends the classification result to the CPr (T_3) and sends 3 types of alerts according to the prediction results: T_6 for the low alert, T_7 for the medium alert and T_8 for the high alert.

The last service is that of the App (Fig. 1(e)). The set of its internal operations are the notification (T_9) and the application to register (T_1) which allows

a new patient to deposit his information. After registration, the provided information will be sent (T_2) to the CPr. The App shares patient information with the Fog ($T_3!$ & $T_4?$) when this latter communicates for the first time with the Br. It also shares the medical history with the CPr ($T_5!$ & T_6) when it fails to find the patient ID. At the end, the App receives two types of alerts ($T_7?$ for the medium and $T_8?$ for the high) when the risk of a heart attack is detected. The App must be opaque with regards to its set of secret states when dealing with either the CPr or the Fog. To match these needs the observable transitions are $\Sigma_o = \{T_2?, T_3!, T_4?, T_5!, T_6?, T_7!, T_8!\}$, while the unobservable ones are $\Sigma_u = \{T_1, T_9\}$. The set of secret states are $S = \{S_2, S_6, S_7, S_9, S_{10}, S_{11}\}$, with S_2 is related to the request to register a patient, S_6 is that related to sending patient data, S_7 is that triggered due to the sending of personal information of a new patient, S_{11} is related to sending the medical history, and finally S_9 and S_{10} reflect the secrets associated with sending the notification. Conducting the opacity verification, we obtain the SOG depicted in Fig. 2(c). We say that the App SOG is not opaque, and it is not K -step weakly, and strongly opaque.

5 SOG-Based Enforcement of Opacity

In this section, we describe the opacity enforcement problem introducing algorithms to secure the heart attack detection system. Considering a language L and a secret language $L(\varphi) \in L$, when opacity fails of a secret φ for a finite system S , we provide an effective method to synthesize automatically a system S' obtained by minimally modifying the system S so that the secret φ is opaque for S' . To synthesize S' , we focus on language modification. If a secret language $L(\varphi)$ is not opaque for a system behavior described by the language $L(S)$, we can modify the behavior by padding it with dummy behaviors. We can then extend the language by computing a minimal super-language of L . In [15], the author has derived an algorithm to compute $\min \prod_{super}^\varphi$ to assist the designer develop a system that satisfies the opacity property for a secret language.

Theorem 1. [15] *Let a language L defined on an alphabet $\Sigma = \Sigma_o \cup \Sigma_u$ and a static projection π_O defined above on the same alphabet and a secret $\varphi \subseteq L$, then:*

$$\min \prod_{super}^\varphi(L) = L \cup (\pi_o(\varphi) \setminus (\pi_o(\varphi) \cap \pi_o(L \setminus \varphi)))$$

The proposed approach builds upon the SOG structure to check the system's opacity. If the system is not opaque, the SOG construction allows for detection of all opacity violations provided as a counterexample. These counterexamples will later be used to improve the system security (opacity) by locating the paths leading to the disclosure of private information and performing necessary changes that would render it opaque. Then we compute the minimal super-language that provides us with the restricted language to be added in order to modify the system behavior. For each incident of opacity violation, we match a trace among the calculated super-language and an unobservable event will be added to this trace. In order to opacify the system, we apply the backtracking method. We implement adjustments where needed to the SOG and the LTS and we thus return to the starting model, the Petri net.

5.1 The SOG-Based Algorithm for the Verification of Simple Opacity

The use of SOG-based algorithm in the verification of simple opacity proved efficient [3]. This is due to the symbolic representation of the aggregates, and to the on-the-fly verification. The SOG construction is stopped when the property is proven unsatisfied and a trace (counterexample) that violates the opacity is supplied. To adopt this algorithm for our enforcement approach, we will bring necessary modifications to it.

Algorithm 1. SOG-based Opacification

```

Procedure: SOG-based Opacification
     $((P, T, F, W), m_o, m_S, \Sigma_o \cup \Sigma_u)$ 
1 Vertices  $V$ ; Edges  $E$ ;
2 Aggregate  $a, a'$ ;
3 Stack  $st$ , CounterExample;
4 Incidence Matrix  $C$ ;
5 begin
6    $(Q, q_{init}, \Sigma, \delta) \leftarrow$  BuildReachabilityGraph( $P, T, F, W, m_o$ );
7    $S \leftarrow m_S$ ;
8    $a \leftarrow Saturate(\{q_{init}\})$ ;
9   if ( $a \subseteq S$ ) then
10  |  $CounterExample.Push(\epsilon, a, \epsilon, a)$ ;
11  end
12   $V \leftarrow a$ ;  $E \leftarrow \emptyset$ ;
13   $trace \leftarrow \emptyset$ ;
14   $st.push((a, EnableObs(a)))$ ;
15  while ( $st \neq \emptyset$ ) do
16  |  $(a, enb) \leftarrow st.Top()$ ;
17  | if ( $enb \neq \emptyset$ ) then
18  | |  $st.Pop()$ ;
19  | else
20  | |  $t \leftarrow RemoveLast(st.Top.Second())$ ;
21  | |  $a' \leftarrow Img(a, t)$ ;
22  | |  $a' \leftarrow Saturate(a')$ ;
23  | | if ( $Treated(a')$ ) then
24  | | |  $E \leftarrow E \cup t$ ;
25  | | |  $Save(a \xrightarrow{t} a')$ ;
26  | | else
27  | | | if ( $a' \subseteq S$ ) then
28  | | | |  $Trace = Print$ 
29  | | | |  $CounterExample()$ ;
30  | | | end
31  | | |  $V \leftarrow V \cup \{a'\}$ ;
32  | | |  $E \leftarrow E \cup t$ ;
33  | | |  $Save(a \xrightarrow{t} a')$ ;
34  | | |  $st.Push(a', EnableObs(a'))$ ;
35  | end
36  end
37  end
38  if ( $CounterExample \neq \emptyset$ ) then
39  | Opacification();
40  end
41 end
```

Algorithm 2. Opacification

```

Procedure: Opacification()
1 begin
2    $minSL =$  ComputationMinSL( $L$ );
3   while ( $CounterExample \neq \emptyset$ ) do
4      $(trace, a, t, a') \leftarrow CounterExample.Top()$ ;
5     if ( $NotTreated(a')$ ) then
6       foreach  $u$  in  $minSL$  do
7         if ( $u = trace$ ) then
8           /* SOG
9             Opacification
10            */
11            $q_{new} = newState();$ 
12            $a' \leftarrow a' \cup \{q_{new}\};$ 
13            $Save(a \xrightarrow{t} a')$ ;
14           /* LTS
15             Opacification
16             */
17            $q \leftarrow CounterExample.Top.Fourth();$ 
18            $t_{new} \leftarrow newUnobservableTransition();$ 
19            $Q \leftarrow Q \cup q_{new};$ 
20            $\Sigma_u \leftarrow \Sigma_u \cup t_{new};$ 
21            $\delta(q, t_{new}) = q_{new};$ 
22           /* Petri net
23             Opacification
24             */
25            $p_{new} \leftarrow newPlace();$ 
26            $P \leftarrow P \cup p_{new};$ 
27            $T \leftarrow T \cup t_{new};$ 
28            $p \leftarrow getPlace();$ 
29            $F \leftarrow F \cup (p, t_{new});$ 
30            $F \leftarrow F \cup (t_{new}, p_{new});$ 
31            $W \leftarrow W \cup \{( (p, t_{new}) \longmapsto 1 ), ((t_{new}, p_{new}) \longmapsto 1 ) \};$ 
32            $C(p_{new}, t_{new}) \leftarrow W(t_{new}, p_{new}) - W(p_{new}, t_{new});$ 
33         end
34       end
35     end
36   end
37   if ( $CounterExample \neq \emptyset$ ) then
38     Opacification();
39   end
40   CounterExample.Pop();
41 end
```

Taking into account that we are trying to opacify Petri nets, the first modification needed to the algorithm presented in [3] consists in replacing the input by a Petri net-modeled system. The petri net has 2 sets of transitions: observable and unobservable actions, and a set of secret marking subsequently representing the states judged to be secret in the LTS. We add in line 3 a Stack, namely *CounterExample* with all the standard functions (*push*, *pop* and *top*), whose elements are quadruples composed by the counter-examples, a transition t , an actual aggregate a and an aggregate a' , successor of a by t . Then, the algorithm 1 starts by constructing (line 6) the reachability graph which represents the LTS. Once other changes have been made (i.e. line 10 & 29), when the opacity is violated, neither the verification nor the construction of the SOG stops. All the paths leading to the disclosure of privacy are stacked into *CounterExample*. Once all nodes are explored and the SOG construction is finished, and if the stack is not empty we proceed to opacification.

5.2 The Opacification Proposed Algorithm

The opacification algorithm has a pretty straightforward mechanism. It begins by computing the minimal super-language. The next step consists in recuperating (line 4) the first elements of the stack (*CounterExample*). Next, the algorithm goes through the *foreach* loop which takes each word of the calculated super-language. If such a word is equivalent with the trace recuperated from the stack, then we proceed to opacify the SOG. We begin by creating (line 8) a new state q_{new} that we will add (line 9) into the aggregate a' . At line 11, we pass to opacify the LTS. We retrieve the last state q included in the aggregate a' . A new unobservable transition t_{new} will be created. Then, the algorithm inserts (line 13) the new state q_{new} to the LTS states, adds (line 14) the new transition t_{new} to the set of unobservable events Σ_u , and defines the transition function between q , t_{new} and q_{new} . Starting from line 16, the algorithm performs the Petri net opacification by creating, at first a new place p_{new} and adding it to the set of places. It also adds the transition t_{new} to the set of transitions. To specify the flow relation between p , t_{new} and p_{new} , the algorithm adds an arc for each relation and assigns to each arc a weight. Afterwards, it modifies the incidence matrix. Finally, the algorithm pops the stack and restarts the operations until the final emptying of the stack presenting the ending test of the while loop.

Being a particular type of Petri nets, oWF-nets require different method of opacification. When fetching the place p (the execution of *getPlace*), we have to exclude the output places. Furthermore, oWF-nets require only one final place p_o . So, following the addition of the unobservable transition t_{new} , we must escape adding the new place. And a flow relation will be added between t_{new} and p_o . Other specific case that may be necessary, when the place returned by *getPlace* is a destination place, we require further changes on the oWF-net. The first step is to retrieve the transition that following its crossing marked the output place. Step two is to delete the flow relation between t and p_o . The following step is to create a new place p_{new} and to add the unobservable transition t_{new} . Then, we

create the flow relations between t , p_{new} , t_{new} and p_o . For the application of the opacification function on our case study, see in this paper [17].

6 Conclusion and Future Work

In this paper, we used opacity, a generalization of many security properties, as a means to track the information flow in an IoT-based medical application. We introduced a model to analyze the behavior of an IoT-based heart attack detection system discussing how an observer may infer personal patient information. Our work aims at detecting security leaks, using SOG-based algorithms for the on-the-fly verification of opacity variants (simple, K -step weak, and K -step strong opacity). We have also proposed a novel, SOG-based approach for opacity enforcement of Petri net-modeled systems. The main contribution of this work is to propose an efficient algorithm for enforcing simple opacity by padding the system with minimal dummy behavior. In our future research, we will explore the same idea of enforcement for other opacity variants such as K -step weak and K -step strong opacity. We also hope to extend this work to take into account different types of enforcement, such as supervisory control for opacity and finding the supremal sub-language, instead of computing the minimal super-language.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
2. Bonomi, F., Milito, R.A., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC@SIGCOMM 2012, Helsinki, Finland, August 17, 2012. pp. 13–16 (2012). <https://doi.org/10.1145/2342509.2342513>
3. Bourouis, A., Klai, K., El Touati, Y., Ben Hadj-Alouane, N.: Checking opacity of vulnerable critical systems on-the-fly. *Int. J. Inf. Technol. Web Eng. (IJITWE)* **10**(1), 1–30 (2015)
4. Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to transition systems. *Int. J. Inf. Secur.* **7**, 421–435 (2008)
5. Dubreil, J.: Monitoriting and Supervisory Control for Opacity Properties. Ph.D. thesis, University of Rennes 1, November 2009
6. Falcone, Y., Marchand, H.: Various Notions of Opacity Verified and Enforced at Runtime. Technical report INRIA (2010)
7. Falcone, Y., Marchand, H.: Runtime enforcement of K-setp opacity. In: 52nd IEEE Conference of Decision and Control, pp. 7271–7278, December 2013
8. Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W.: Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Trans. Parallel Distrib. Syst.* **24**(1), 131–143 (2013)
9. Lin, F.: Opacity of discrete event systems and its applications. *Automatica* **47**(3), 496–503 (2011)
10. Lowe, G.: An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.* **56**(3), 131–133 (1995)

11. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the soa. *Ann. Math. Comput. Teleinform.* **1**, 35–43 (2005)
12. Mazaré, L.: Using unification for opacity properties. In: Proceedings of WITS (Workshop on Information Technology and Systems), vol. 4, pp. 165–176 (2004)
13. World Health Organization: May 2014. <http://www.worldlifeexpectancy.com/tunisia-coronary-heart-disease>, consulté le 14/02/2017
14. Saboori, A., Hadjicostis, C.N.: Opacity-enforcing supervisory strategies via state estimator constructions. *IEEE Trans. Automat. Contr.* **57**(5), 1155–1165 (2012). <https://doi.org/10.1109/TAC.2011.2170453>
15. Yeddes, M.: Enforcing opacity with orwellian observation. In: 13th International Workshop on Discrete Event Systems, WODES 2016, Xi'an, China, 30 May–1 June, 2016, pp. 306–312 (2016). <https://doi.org/10.1109/WODES.2016.7497864>
16. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. *J. Internet Serv. Appl.* **1**(1), 7–18 (2010)
17. Zrelli, R., Yeddes, M., Ben Hadj-Alouane, N.: Checking and enforcing security through opacity in healthcare applications (2017)



Power-Based Device Recognition for Occupancy Detection

Azkario Rizky Pratama^{1,2(✉)}, Widyawan², Alexander Lazovik¹,
and Marco Aiello¹

¹ Distributed Systems Group, Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, Groningen, The Netherlands
a.r.pratama,a.lazovik,m.aiello}@rug.nl

² Department of Electrical Engineering and Information Technology,
Universitas Gadjah Mada, Yogyakarta, Indonesia
widyawan@ugm.ac.id

Abstract. Each person using electrical devices leaves electricity fingerprints in the form of power consumption. These can be very useful for understanding the context of that person in, for instance, a smart office. A device that is highly correlated with the presence of a person in an office is the computer monitor; the correlation is in the range 83–96%. Therefore, it is useful to recognize from an aggregated power load the portion that is due to computer monitors. In this paper, we propose an event-based device recognition approach. After studying several predictors and features for device classification, we build a prototype for the classification. We evaluate the approach with actual power measurement of seven office monitors used by four workers in an office environment. Our experiments show that the approach is feasible and the per-day accuracy ranges in the range 69–80% for seven and five physical devices, respectively.

Keywords: Device recognition · Load disaggregation
Occupancy detection · Appliance recognition

1 Introduction

Smart buildings operate efficiently by being aware of their actual use and environmental conditions [1]. One of the biggest challenges to achieve smart buildings is the automatic classification of human activities and state within the building. Low intrusive approaches are generally preferred due to privacy and economic concerns [2]. The aggregated measurement of power consumption is an important feature to consider for context mining and human activity classification.

Human presence detection is one of the necessary components in a smart building system to provide a custom service, specific to present occupants. Some common examples are automated lighting and HVAC (heating, ventilation, and cooling) systems that automatically tune their operations on the users' occupancy [3]. These systems require the environment contexts (such as human presence) to be updated accordingly. Since the context of the building is highly

dynamic, we need to keep the building systems up to date, in order to assure composed services achieve building operation objectives, such as providing occupant satisfaction and efficient energy consumption. Furthermore, for privacy and performance reasons, the processing of such context should happen at the edge of the network, before further sending selected data and knowledge to cloud infrastructures. Such a *fog computing* paradigm supports the demands of quick and efficient data processing while having connected service-based systems. In the present work, we do not focus on the service-oriented architecture and cloud components, as they are rather standard, while instead we focus on the IoT and Smart Building aspects. In particular, we investigate the feasibility of load disaggregation from a unique power consumption reading with the final goal of correctly classifying the human presence in an office. We collect large amounts of data using a global power meter/smart meter. These meters are increasingly installed by utilities, are relatively inexpensive, and do provide basic energy readings with reasonable sampling rates. Such an approach opens the possibility of recognizing personal occupancy using global room-level or department-level meters. In other terms, by recognizing particular devices associated to a particular person, we obtain the reduced-size, finer-grain occupancy information which further can be forwarded to the cloud for capturing the bigger picture about building occupancy. The chosen office devices for the present study are computer monitors as there is evidence that most of the time people are in offices they are engaged in computer related activities. E.g., in the US, workers spend on average of more than six hours per day at the computer and an additional hour at home [4]. A first indication that the computer use is closely related to office presence and work.

To learn which approach works best, what values to provide to our models, and to evaluate the performance of the approach, we experimented for two and a half months in our own offices at the University of Groningen. We collected power consumption data using global power meters. We deployed power meters in a single point measurement in incoming electrical line as well as per-appliance plugs to collect ground truth information and observe characteristics of every monitor screen. To make the approach more flexible and portable, we define synthetic aggregate data by applying superposition of several monitors that are owned by the same person in an office. The rationale for this choice is that no significant differences are found between the synthetic signals and the composite loads measurement on the electrical line. From each device, we extract and explore several features that possibly characterize turning on/off events. Such descriptors are used to train classification models to infer which are the active devices at any time. We develop a device recognition approach which is based on event detection. Events are triggered when potential switching occurs.

The contribution of the present work can be summarized as follows:

1. We propose a novel device recognition system (specifically, computer monitors) based on energy load disaggregation;
2. We compare and identify meaningful features to describe switching events of monitor instances recovered from a single electricity measurement (i.e. active power);
3. We evaluate experimentally the approach; and
4. We propose the concept of *virtualdevice* to detect multi appliances running simultaneously and improve the recognition performance.

The remainder of the paper is organized as follows. The overview of previous, related work is provided in Sect. 2. We describe the system’s design and implementation in Sect. 3. Experimental setup and evaluation are provided in Sect. 4. In Sect. 5, experimental results are reported and discussed. Finally, conclusions are drawn in Sect. 6.

2 Related Work

Diverse sensor types have been used to obtain occupancy context both in residential and commercial buildings. These include RFID, passive infrared PIRs, door sensors, GPS, WiFi, temperature, humidity, and other environmental sensors [3, 5, 6]. Binary occupancy detection, based on electricity consumption has also been proposed, e.g. [7, 8]. Lu *et al.* make use of historical occupancy and indication of human presence (through PIR and door sensors) to infer occupancy states in homes [3]. By using a Hidden Markov Model, they show that 88% occupancy accuracy can be achieved. In this work, we aim to address the more general case of multiple people detection, rather than individual home occupancy inference.

Load disaggregation, also referred to as device separation, deals with the identification of consumptions of individual devices from a global electricity consumption signal. The most common approach is based on recognizing appliance signatures. Liang *et al.* define two signature forms [9]. First, a signature can be recognized in *snapshot* form: an observation of load behavior at any fixed time intervals. Second, the signature can be formed as *delta* form: taking parameter changes into account when a state transition occurs.

In [10], the authors observe appliance switching events to learn characteristic of several devices in an office, such as monitors and printers. They observe and describe the behavior of several type of monitors, without trying to supervise models and classify the fresh data. Low power appliances recognition using 120 changing states of appliances is presented in [11]. The performance shown is 90% and 76% for individual and multiple appliance recognition, respectively. To achieve these results feature-rich, high-resolution power meters were employed. Due to National regulations on smart meters [12], to keep the study realistic with standard installations, in the present study we decide to utilize generic power meters with only active power measurement capability.

An effort to classify personal occupancy in the office was developed by [13]. The authors deploy one power measurement on each work desk and classify whether the respective occupant is present, away, or in standby. By using two weeks worth of data, they show 93% accuracy with a KNN-based classification method.

3 Design and Implementation

Smart and energy-aware buildings [1], rely on a number of components, such as a context inference and repository component, an AI Planning and Scheduling one, and an orchestrator one [14]. Figure 1 shows an architecture derived from our previous work. The present contribution aims at offering improved context information which is in turn essential to determine the current state of the building. The context is inferred on-site to enable efficient data processing and reduce the amount of data to be transported to the cloud. The current state of the building, possibly with the bigger picture of how the persons occupy the building (e.g. processed in the cloud), affect the plan composing. The plans then go to the orchestrator which is responsible for the actuation and for evaluating possible failures or execution deviations, in turn affecting the context again.

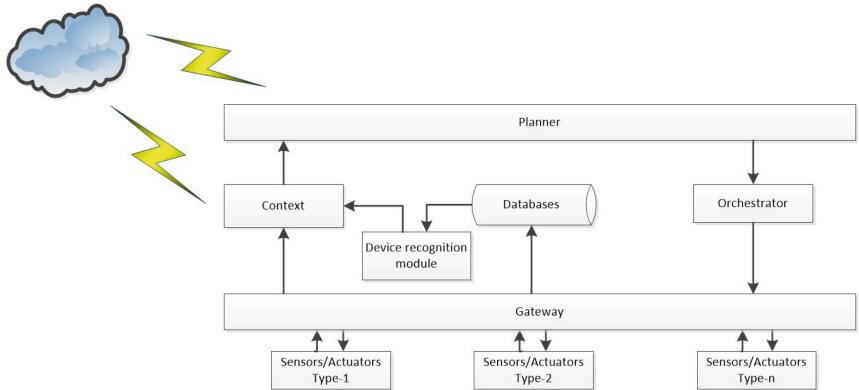


Fig. 1. Software architecture of a smart building, derived from [14].

The context is a point in a possibly infinite feature space of relevant building context variables. The move from one point to another one is defined as an *event*. In the specific case of load disaggregation, an event is a significant peak or slope occurring in power consumption waveforms. These are typically associated with a device (electricity load) switch going from ON to OFF, or vice-versa. We develop a mechanism to detect candidate events using thresholds and validate them according to empirical data (i.e., 10 W difference, 60 s between two consecutive events). These values are derived from previous experimentation. In particular,

the watt-difference parameter is based on the study of monitors of different brands and types with the lowest power consumption for them positioned at 12 W. The time interval parameter is based on the fact that people typically work in burst higher than one minute. The precise processing is presented as Algorithm 1.

Algorithm 1. Event detection from an aggregated data

```

1: procedure EVENT DETECTION AND EVENT VALIDATION
2:    $X \leftarrow \text{aggregated data}$ 
3:   set window in moving windows
4:   get events:
5:     compute range in a window
6:     if range > wattThreshold then
7:       if durationBetweenEvent > durationThreshold then
8:         event  $\leftarrow$  window
9:   get delta power:
10:    compute mean after and before an event
11:     $\Delta P \leftarrow (\text{mean}_{\text{after}} - \text{mean}_{\text{before}})$ 
12:   validate events:
13:   if  $\Delta P > \text{wattThreshold}$  then
14:     validatedEvent  $\leftarrow$  event

```

For each combination of validated events, we extract the relevant features. Inspired by the field of dynamic systems [15] and statistics [8], we consider: rise-time, overshoot, steady level, variable variance, and mean of absolute difference, described as follows.

Delta P (ΔP) is the difference of average power before a detected event and average power after the event. We consider five samples for both before and after events; illustrated as black arrows in Fig. 2.

Steady level is the value of a device (or set of devices) in a stable state; represented as dashed line in Fig. 2.

Rise time is the time needed for a transition from 10% to 90% of the reference levels; represented as a grey shaded rectangle in Fig. 2.

Overshoot is the percentage of the difference between state levels. It is defined as Eq. 1, where y_{max} is the maximum value (indicated by downward-pointing triangle in Fig. 2), $level(s_k)$ is the steady state level, and $|A|$ is the amplitude [16].

$$\text{Overshoot} = \frac{(y_{max} - level(s_k))}{|A|} 100\% \quad (1)$$

Mean of Absolute Difference (MAD) captures the ripples during a device's active period, Eq. 2 [8].

$$MAD = \frac{1}{N-1} \sum_{i=2}^N |y_i - y_{(i-1)}| \quad (2)$$

Variance measures how far a set of values are spread out from the steady level, i.e.:

$$var = \frac{1}{N-1} \sum_{i=1}^N |y_i - \bar{y}|^2 \quad (3)$$

Features such as Power Level, MAD, and Variance satisfy the criterion of so-called additive features [9], therefore it is possible to compute the delta value of an event; see Algorithm 2.

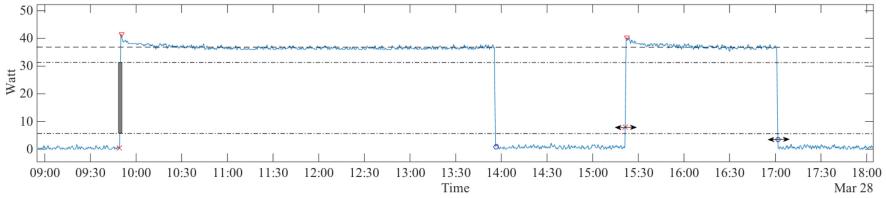


Fig. 2. Example of a day worth of feature values

Algorithm 2. Feature extraction from aggregated data

```

1: procedure FEATURE EXTRACTION
2: get features between validatedEvents:
3:   for all combination validatedEvents do
4:     compute {RiseTime; Overshoot; Level; Peak; MAD; Variance};
5:   get delta features
6:    $\Delta_{features} \leftarrow (Level_t; MAD_t; Var_t)_{t-1}^t;$ 

```

The extracted features contribute to the classification of the event and new context state. Several classification methods are possible:

k-Nearest Neighbor is one of the simplest learning techniques that works by finding the predefined number of labeled samples nearest to a query and predict the class label with the highest votes [17].

Naive Bayesian is a simple probabilistic classifier that assumes features are independent given a class label [18]. We choose this technique with an assumption that each feature contributes independently to the probability of class labels, regardless of any correlations between the features.

Neural network is a nonlinear statistical model for regression or classification, typically represented by a network diagram [19]. It works by deriving hidden features Z from linear combination of the inputs X and then modeling the target classification Y as a function of linear combination of the Z.

Due to the flexibility of the input features and the easy extensibility of the network structure, we choose single layer Neural networks and extend to multiple layers, as illustrated in Fig. 3.

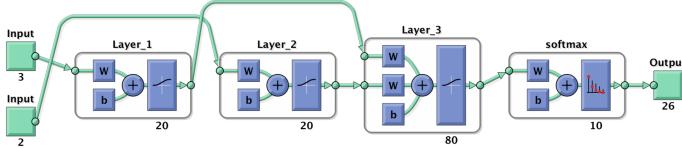


Fig. 3. Multiple hidden layer, with two inputs and three hidden layers.

We further define *virtualdevice* as a combination of two or more physical devices belonging to a specific person. *virtualdevice* are useful when the composing devices change their state concurrently. In the present setup, *virtualdevice* is denoted by device index number 24, 25, and 26.

4 Evaluation

To evaluate the approach, we experiment in our own offices located in Groningen on the fifth floor of the Bernoulli building on the Zernike Campus of the University of Groningen. The experiment took place from the 13th to the 31st of March 2017 and from the 17th April to the 22nd June 2017.

Setup. We consider two office rooms occupied by four people (PhD students). To collect the ground truth, we equip all electric devices of the rooms with Plugwise Circle, the single power consumption sensors from Plugwise¹. Each Circle utilizes the wireless ZigBee protocol. We use Raspberry Pi² to pool the data from the plugs and forward them to a server for processing.

We sample data at 10 s intervals to assure there is enough time for the pooler to receive data from all plugs. Furthermore, this value is set to comply with the Dutch National regulation on smart meters [12]. If due to some failure, we miss a reading, we keep the previous valid one. This approach is common to mimic the constant consumption of simple devices, such as LCD monitors [13]. We then analyze the recorded data to attest the system performance. From each individual power load, we extract features of events for teaching learning models and construct ground truths from known switching events. We supply two weeks data to train models and use two months fresh data to examine the classification performance of the models. The details of number training and testing set is summarized in Table 1. From the table there is an indication that the number of training instances depends on the considered training labels. It can also be seen that the number of test data relies on target devices. The more target devices are included, the more frequent occupant presence should be detected. Thus the number of available test instances is also increased.

The actual presence of people to populate the ground truth is taken manually, based on paper based diary and human observation.

¹ <https://www.plugwise.com>.

² <https://www.raspberrypi.org/>.

Table 1. Summary of number of training and testing set

No	#instances		#training-labels	Target devices
	Training	Test		
1	252	78 (10 days)	8	[4;10]
2	252	160 (27 days)	8	[4;7;10]
3	252	188 (27 days)	8	[4;5;7;10]
4	252	298 (31 days)	8	[4;5;7;10;14]
5	252	274 (31 days)	8	[4;5;7;10;14;(24)]
6	317	241 (31 days)	10	[4;5;7;8;10;11;14;(24;25;26)]

Metrics: Event Detection. To evaluate the experiment, we measure how accurate the proposed approach is in classification. We resort to standard metrics, such as *Precision*, the rate of True Positive over all events detected by system regardless the truth, and *Sensitivity*, the proportion of real events that are correctly identified.

Metrics: Device Classification. The precision of classification is defined on the basis of the actual belonging of devices identified by the k *most-probable* classe to the correct class. We use the average of how many classes are correctly inferred, Eq. 4, using $k = 2$ with an constant weight.

$$\text{Accuracy}_{\text{per day}} = \frac{1}{n_{\text{events}}} \sum_{i=1}^{n_{\text{events}}} (\hat{y}_1 \cap \hat{y}_2) \quad (4)$$

The average, overall accuracy, can then be computed as Eq. 5:

$$\text{Accuracy}_{\text{average}} = \frac{1}{d_{\text{days}}} \sum_{d=1}^{d_{\text{days}}} \text{accuracy}_{\text{day } d} \quad (5)$$

5 Results and Discussion

Following (Algorithm 1), we perform device detection on the acquired data set. The result is shown in Table 2. The devices mentioned in the table are monitor screens (i.e., monitor 4 and 5; 7; 10; and 14 belong to Worker W1, W2, W3, and W4, respectively) and a *virtualdevice* (i.e., device 24 which is a combination of device 4 and 5). The number of days shows how many days these devices are used or activated during observation. The best performance of event detection is when the system only detects two devices. The precision and sensitivity reaches 87.9% and 97.3%, respectively. As the number of involved devices increases, the performance declines, reaching 70% precision and 89% sensitivity. A significant

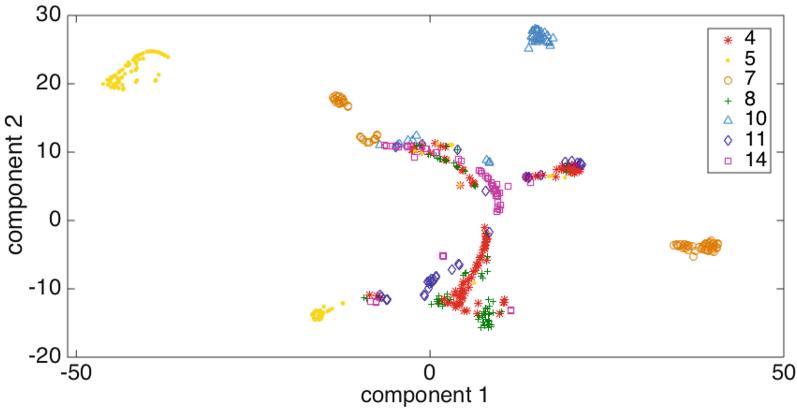


Fig. 4. Visualization of seven device classes.

drop occurs when device 14 is added to the aggregated power, while adding other devices gradually changes the performance by just 1%. Device 14 worsen the overall event detection. The reason for this is in the short interval transitions that occur in the dataset for this device (i.e., 2 consequent transitions, ON-OFF-ON, in less than 5 min), thus resulting in possible undetected events.

Table 2. Device events detection.

No	#days	Devices	Precision	Sensitivity
1	10	[4;10]	0.879	0.97333
2	27	[4;7;10]	0.87361	0.94193
3	31	[4;7;10;14]	0.71002	0.92854
4	31	[4;5;7;10;14]	0.71575	0.85877
5	31	[4;5;7;10;14;(24)]	0.70157	0.8912

Events-extracted features can be visualized using t-Distributed Stochastic Neighbor Embedding, Fig. 4 [20]. The features taken into account for the analysis are ΔP , steady level, rise time, overshoot, mean of absolute difference, and variance. One can observe the challenge of device classification; in fact, classes are not easily separable.

We compare several feature sets and three different techniques to classify particular devices. We also observe the significance of number of recognized devices, starting from two (devices 4 and 10) and up to six devices (physical devices 4,5,7,10,14, and virtual device 24). The comparisons are summarized in Fig. 5.

Feature set 1 considers the difference in power before and after an event (ΔP). The accuracy is 32% and 45% using NeuralNet and NB techniques, respectively.

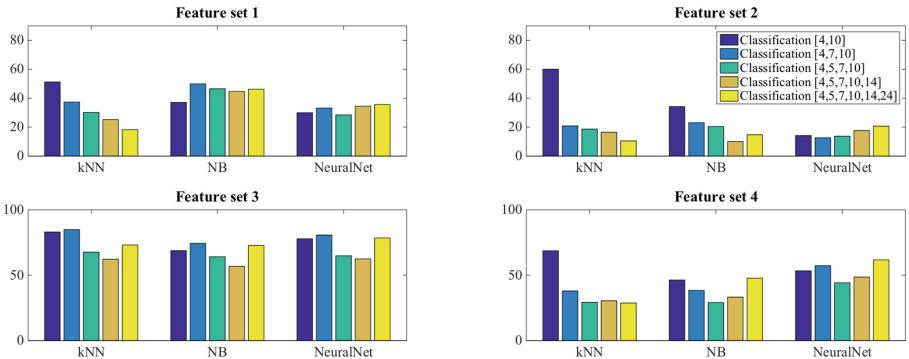


Fig. 5. Comparison of feature sets.

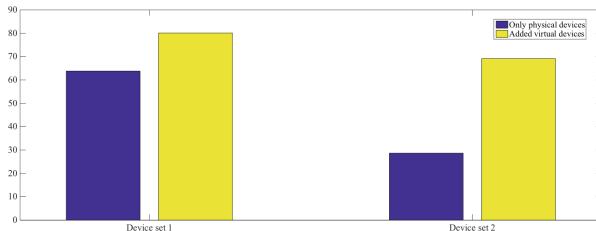
For these methods, the performance seems not to be affected by the number of devices. By using the same feature, the accuracy with kNN reaches 50%. However, increasing the number of appliances does result in considerably decreasing performance using kNN.

The rise time and overshoot in Feature set 2 also give fluctuations in terms of per-day accuracy, depending on the number of devices to be classified, i.e., 15–21%; 10–34%; and 10–60% for NeuralNet, NB, kNN, respectively. This feature set brings the lowest performance compared to the other sets. It is also shown that the performance of kNN and NB method depends on the number of device. The higher number of devices considered, the less performance can be achieved. With respect to NeuralNet, it shows stable results below 21% accuracy with this feature set.

The combination of steady-level, MAD, and Variance in Feature set 3 delivers the highest performance among the other feature combinations, up to 81%, 74%, and 84% for the NeuralNet, NB, and kNN, respectively. It is worth noting that these results would degrade as the number of classification device increases, reaching 62% for both NeuralNet and kNN, and 56% for NB. Our proposed concept of *virtualdevice* can improve the performance by 16%, 16%, and 11% for NeuralNet, NB, and kNN, respectively, by introducing *virtualdevice* to the classification models. Such improvements are presented as an upward trend from the 4th- to 5th-bar for three models in Feature set 3, in Fig. 5.

Feature set 4—a combination of Feature set 2 and 3—does not deliver a better result than the others. However, the average performance of NN outperforms NB and kNN in recognizing of 6 devices by 14% and 33%, respectively. This result is in accordance with the result of experiment with feature set 3 where the NeuralNet slightly outperforms the others.

In addition to physical devices, it is useful to also consider virtual ones, resulting from the combination of measurements from physical ones. In the evaluation, let us consider seven physical devices owned by four people (three have multiple screens). By introducing three virtual devices (i.e., 24, 25, 26), we classify these

**Fig. 6.** Simultaneous devices classification

devices with Feature set 4 in a modified network structure, as shown in Fig. 3. The device recognition result is shown in Fig. 6.

Device set 1 that consists of five physical devices can be recognized correctly with 63.81% accuracy per-day. By adding one virtual device that represents two devices activated almost simultaneously, 80.1% accuracy per-day is achieved. The accuracy of recognizing device set 2 with seven physical devices drops significantly to 28.65% accuracy per-day. By introducing three virtual devices represents six devices, the performance improves, reaching 69.13% accuracy per-day.

5.1 Relation of Monitor Screen Activation and Occupancy

The observation of occupancy during working hours (set from 8 am to 9 pm, due to the different working times of individuals) is shown in Table 3. The monitors reveal the accurate occupancy of people up to 96.8%. It is lower, about 83.5%, for the person who is present at the office for 4 days of a week observation. The reason is that the monitor needs to wait its timeout to automatically put on sleep mode after plugged out from the laptop/sources. This does not happen to worker W1 and W2 due to different hardware specifications.

Table 3. Occupancy accuracy over a 5-min interval

Worker	Presence days	Accuracy
W1	7d	96.7949
W2	5d	89.8718
W5	4d	83.4936

5.2 Discussion

Based on the evaluation in an actual office space, we conclude that the proposed event detection approach has very promising performance. It achieves 90% sensitivity with a 70% precision. In other words, the system is good at the detection of the actual events, yet of all inferred events, some are misread. In fact, the

system misinterprets oscillations on the waveform as switching events. This happens as the considered devices have a low-power consumption, making harder to discern the events from common, regular fluctuations.

The Feature set 2 (rise time and overshoot) are not describing the devices very well. The reason could be in the time required for the positive-going transition not being captured by the 10s data sampling. On the contrary, Feature set 3 shows the best performance among the others. This set is applicable to the three methods with comparable results, up to 84% accuracy per-day. The combination of Feature set 2 and 3 does not contribute to improving the performance of kNN, NB, and NeuralNet. However, with the same features, the classification works better in multiple hidden layers network (Fig. 3) than in single hidden layer network.

The classification performance of kNN suffers from the dependencies of the number of devices. It can be moderately dropped as a number of considered devices increase. Conversely, the performance of NeuralNet and NB is more robust to the number of involved devices. This is because kNN works by finding the nearest labeled sample. As the sample of training set during 2 weeks does not cover very well for all devices, the results of kNN become worse.

Based on our empirical observation during a week, the personal occupancy classification shows acceptable performance in relation to the monitor activation. However, some factors might affect the relation, such as whether the person is working using electrical devices, a personal habit to consistently deactivate the device while being away, and hardware configuration (auto sleep mode).

6 Concluding Remarks

Even with simple aggregated power consumption, it is possible to recognize device usage and turn that information into building-user context knowledge. We have proposed an approach based on neural networks and evaluated over ten days in an actual office. We used various power features, such as ΔP , steady power level, rise time, overshoot, MAD, and variance.

The experimental evaluation shows that it is possible to recognize low-power devices from composite energy loads, achieving 84%, 81%, and 74% accuracy per-day using kNN, NeuralNet, and NB, respectively. The kNN performance will show a downward trend as the number of devices increased, while NeuralNet and NB seem more robust in the addition a number of devices. We notice that steady-level, MAD, and Variance features give a good description to the classifiers while adding rise time and overshoot features not always give a positive impact. It is also validated that the proposed *virtualdevice* can improve the performance by 40% in the recognition of 10 classes (7 physical devices and 3 virtual devices), reaching 69.13%.

Acknowledgement. Azkario Rizky Pratama is supported by the Indonesia Endowment Fund for Education (LPDP). This research has been partially sponsored by the EU H2020 FIRST project, Grant No. 734599, FIRST: vF Interoperation suppoRting buSiness innovaTion.

References

1. Nguyen, T.A., Aiello, M.: Energy intelligent buildings based on user activity: a survey. *Energy Build.* **56**, 244–257 (2013)
2. Nguyen, T., Aiello, M.: Beyond indoor presence monitoring with simple sensors, pp. 5–14 (2012)
3. Lu, J., Sookoor, T., Srinivasan, V., Gao, G., Holben, B., Stankovic, J., Field, E., Whitehouse, K.: The smart thermostat: Using occupancy sensors to save energy in homes. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys 2010, pp. 211–224 (2010)
4. Microsoft: Us workers spend 7 hours on the computer a day on average (2013). <https://www.onmsft.com/news/microsoft-us-workers-spend-7-hours-computer-day-average>. Accessed 26 Sep 2017
5. Scott, J., Bernheim Brush, A., Krumm, J., Meyers, B., Hazas, M., Hodges, S., Villar, N.: Preheat: controlling home heating using occupancy prediction. In: Proceedings of the 13th International Conference on Ubiquitous Computing, UbiComp, pp. 281–290 (2011)
6. Koehler, C., Ziebart, B.D., Mankoff, J., Dey, A.K.: Therml: occupancy prediction for thermostat control. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp, pp. 103–112 (2013)
7. Mamidi, S., Chang, Y.-H., Maheswaran, R.: Improving building energy efficiency with a network of sensing, learning and prediction agents. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, vol. 1, pp. 45–52 (2012)
8. Jin, M., Jia, R., Spanos, C.: Virtual occupancy sensing: using smart meters to indicate your presence. *IEEE Trans. Mob. Comput.*, **16**(11) (2017)
9. Liang, J., Ng, S.K.K., Kendall, G., Cheng, J.W.M.: Load signature study—Part I: basic concept, structure, and methodology. *IEEE Trans. Power Deliv.* **25**, 551–560 (2010)
10. Kalluri, B., Kondepudi, S., Wei, K.H., Wai, T.K., Kamilaris, A.: Opld: towards improved non-intrusive office plug load disaggregation. In: 2015 IEEE International Conference on Building Efficiency and Sustainable Technologies, pp. 56–61 (2015)
11. Zoha, A., Gluhak, A., Nati, M., Imran, M.A.: Low-power appliance monitoring using factorial hidden Markov models. In: 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 527–532 (2013)
12. P1 companion standard, March 2014. http://www.netbeheernederland.nl/_upload/Files/Slimme_meter_15_32ffe3cc38.pdf. Accessed 08 Feb 2017
13. Akbar, A., Nati, M., Carrez, F., Moessner, K.: Contextual occupancy detection for smart office by pattern recognition of electricity consumption data. In: 2015 IEEE International Conference on Communications (ICC), pp. 561–566 (2015)
14. Georgievski, I., Nguyen, T.A., Nizamic, F., Setz, B., Lazovik, A., Aiello, M.: Planning meets activity recognition: service coordination for intelligent buildings. *Pervasive Mob. Comput.* **38**, 110–139 (2017)
15. Franklin, G.F., Powell, J.D., Emami-Naeini, A.: Feedback Control of Dynamic Systems, 7th edn. Pearson Prentice Hall, Upper Saddle River (2009)
16. IEEE standard for transitions, pulses, and related waveforms - redline. IEEE Std 181–2011 (Revision of IEEE Std 181–2003) - Redline, pp. 1–71 (2011)
17. Shakhnarovich, G., Darrell, T., Indyk, P.: Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. The MIT Press, Cambridge (2006)

18. Friedman, J., Hastie, T., Tibshirani, R.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd edn. Springer, New York (2009)
19. Demuth, H.B., Beale, M.H., De Jess, O., Hagan, M.T.: *Neural Network Design*, 2nd edn. Martin Hagan, USA (2014)
20. van der Maaten, L., Hinton, G., Bengio, Y.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008)



Cognitive Determination of Policies for Data Management in IoT Systems

Aly Megahed^(✉), Samir Tata, and Ahmed Nazeem

IBM Research - Almaden, 650 Harry Rd, San Jose, CA 95120, USA
 {aly.megahed,stata}@us.ibm.com, ahmed.nazeem@ibm.com

Abstract. Internet of Things (IoT) has emerged as a very hot area in the past few years. Managing data in IoT systems is still a challenging research topic, particularly when one aims at determining the correct set of decisions to take given some trigger events in an IoT system. The state of the art in determining such actions and corresponding policies is ad-hoc, based on significant human intervention. In this work, we propose a cognitive automated approach for policy and action determination in IoT systems that uses historical data to learn the best set of actions to take and involves an mathematical optimization module that chooses the optimal set of actions to pursue given the limited resource capacity in the system. Our system requires minimal human intervention and thus could be very beneficial in today's IoT frameworks.

1 Introduction

Internet of Things (IoT) refers to networks of objects, machines, vehicles, and other physical systems with embedded sensing, computing, and communication capabilities [18]. These devices sense and then share real-time information about the physical world. IoT is growing at a highly increasing rate. For example, according to a study by Gartner ([32]), it is believed that, by 2020, there would be 25 Billion connected things. Such connected things produce massive amounts of monitored data produced by sensors and devices. That is why data management has been evolved as one of the recent challenges for IoT systems. Data management policies are one of the main techniques to ensure collecting, managing and disseminating data is performed in a systematic, planned and managed way [37]. Examples of such policies include:

- Applying face detection algorithms to the recorded videos in places where a robbery happened,
- Applying algorithms to focus on car plate recognition when an accident happens in a smart city,
- Calling the fire department when a fire is detected with a detailed request of the number of trucks, appropriate number of fire fighters and required tools.

Depending on the context of the IoT system, data management policies are added to/removed from the IoT system. A naive approach would rely on human

intervention expertise to add/remove policies. Doing so is inefficient and error-prone in many applications. Thus, there is a need for a cognitive system that determines which policies to adopt whenever changes are triggered in the system with as minimal human input as possible. This is what we develop in this paper. That is, we present a cognitive method that automatically determines the appropriate policies in an IoT system, by analytically manage the input data and use historical policies and data to learn appropriate actions.

This paper is organized as follows. First, we present the state of the art in Sect. 2. Then, in Sect. 3, we present our methodology for action recommendation for policy-based data management in IoT systems. We lastly present our conclusions and directions for future work in Sect. 4.

2 State of the Art

Data management in IoT systems typically begins with monitoring the data. There has been multiple works in the literature on data monitoring [2]. Four main monitoring approaches exist, namely sampling and filtering-based monitoring (see, for example, [39]), probing-based monitoring (see, for example, [19, 20]), diagnosis-based monitoring (see, for example, [25, 27]), and performance-based monitoring (see, for example, [10, 14]).

In [39], the authors presented an adaptive monitoring framework for IoT devices. The system adapts the frequency of the monitoring as well as the amount of data going through the system. Their system increases the monitoring frequency when the value of monitored metrics gets to a certain pre-specified threshold and decreases it when a quality of service violation is unlikely to happen. The authors developed a monitoring approach that uses probing and is passive in nature.

In [25], the authors adopted a diagnosis-based monitoring approach based, where system faults are identified and then the metric monitoring is adapted accordingly. In [14], the authors developed a method for integration, validation, and description of metrics of software project performance. Katsaros et al. [21] proposed an architectural approach that combines virtualization and physical levels for monitoring data collection, where many open source solutions were combined (e.g. Lattice [11], Ganglia [23], Nagios [15]) so as to get one holistic application covering different layers. They used collectors for data extraction from different virtual and physical layers, and then data externalization to the upper layer using an external data collector. They also proposed a monitoring manager that acts as an orchestrator of the whole monitoring system. That manager controls and provides the required interfaces to add or consume monitoring information.

In [7], the authors proposed a tool, for distributed systems, for monitoring resources that enhances high-performance computing. The tool extracts both the application and resource state, and assigns new resources and/or shut down unused ones based on these states. This happens during the runtime of the application or in future usages. Data collectors are used to collect the data

from resources and then the data is stored in a distributed database. Later, a statistical analysis is performed to take decisions on the resource assignment in the application, i.e., whether to keep it as the latest assignment or manually reconfigure it.

As for more specific data management literature on IoT, there has been some recent literature. For example, in [40], the authors formulate some design principles for IoT data management. They developed optimization algorithms aimed at producing coherent IoT ecosystems, by means of publish/subscribe middleware and linked data that span over cloud infrastructures and mobile networks. In [13], the authors proposed a distributed data service for data collection and processing in IoT systems. They mentioned that their main goal is to enable multiple and different IoT middleware systems to share common data services that is coming from a loosely-coupled provider. To that end, they specified the corresponding techniques for data collection, filtration, storage, and aggregation to allow for efficient real-time data querying.

Padiya et al. [29] addressed the challenge of handling massive sensor data in an interactive fashion via Resource Description Frameworks (RDF). They compared multiple RDF storage mechanisms such as triple store, vertically partitioned table, horizontally partitioned table, column store, among others. They also represented a set of metrics that were designed to take decision for choosing the appropriate RDF data storage technique *a priori* for IoT systems. In [1], the authors present a survey for data management solutions proposed for IoT or subsystems of IoT. They discussed the different design primitives that they claim to be most important to address in an IoT management solution and showed how they were approached by the proposed solutions. Additionally, they proposed a data management framework for IoT that incorporates the aforementioned design elements and acts as a seed for a comprehensive IoT data management solution.

Gubbi et al. [18] present a cloud centric vision for the world-wide implementation of IoT. They proposed a cloud implementation based on the interaction of private and public clouds and concluded that there is a need for expanding on the convergence of wireless sensor networks. In a previous work [38], we proposed an adaptive monitoring approach in IoT systems, where a metric value-based change or an environmental one triggers the need to choose which metrics to monitor and at what frequency. While an approach for doing this metric monitoring management was presented, no mention of action policies was discussed. A slightly similar work with an application in healthcare can be found in [4].

As one can see, all the previous works dealing with data management in IoT systems do not consider the issue of the determination of policies governing the IoT system. To the best of our knowledge, the only related literature is that concerned with security and control policies, rather than action policies, which is the focus of our work here. For examples on the latter policy focus, we refer the reader to the papers in [33, 36]. It seems that action policies has been done in an ad-hoc and/or manual manner in the current state of the art. Thus, there is a need to fill this gap in the literature, as well as practical systems, via providing

a methodology that handles automated cognitive action recommendation for different events that are monitored within an IoT system. This is what we provide next in this work.

3 Methodology

Figure 1 illustrates the overview of our methodology, where the first step (defined in details in Sect. 3.1) constitutes a recommendation model that gets trained on historical pre-defined/hypothetical events-conditions-actions data. Such a model can then be used for new events-conditions to recommend the actions for such events-conditions. Only in case the recommended actions are not incorporated with a high confidence, they might need some expert validation. The expert validation step is detailed in Sect. 3.2 along with the update of the historical data as per the expert input. We then prioritize the actions recommended by the system in the step detailed in Sect. 3.3 using a model that learns such prioritization from historical data. Lastly, since the system is typically resource-constrained, an optimization mode that chooses the optimal final set of feasible actions to execute, trying to maximize the system utility by giving more resources for the actions with higher priority. We detail this optimization model in Sect. 3.4.

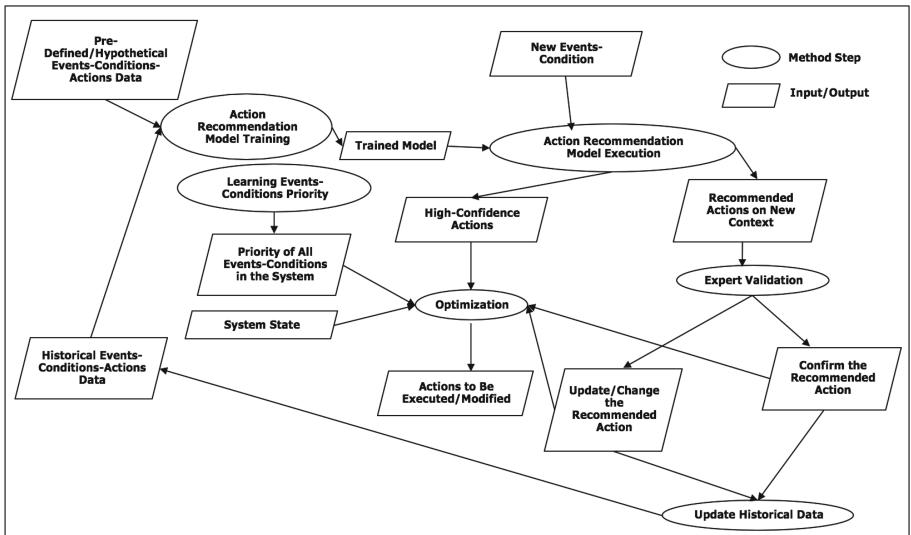


Fig. 1. Overview of our overall methodology

3.1 Recommendation of Action Given an Event-Condition

We assume that we are given a set of event-condition-action triplets. That is, for each given historical or hypothetical event, we are given a condition and a corresponding action. An example of elements of that set in our IoT context would

be a robbery (event) that happens at a bank (condition) with the corresponding actions being calling the police and focusing the cameras of the smart cities on plates of cars surrounding the bank. Now, what we want to achieve in this step of our methodology is that whenever a new event-condition happen, the system is able to automatically recommend the best action to take.

In order to achieve this, we develop a classification model that classifies the events-conditions to the corresponding actions. We refer the reader to [6, 17, 26] for extensive analysis of machine learning classification. We note that the events-conditions are typically text data and thus, in order to do such classification, we need to first perform some text mining in order to retrieve the features of these events-conditions. This can be done as follows: We convert each event-condition to a bag of words. This text is transformed in order to get label indexers. The label indexers fit on the whole dataset. Additionally, we develop feature indexers that automatically identify categorical features and have them indexed. Then, we are ready to build the classification model that gets trained on such historical data to predict the recommended action. Several machine learning techniques can be used for each of the two steps. For the text mining, we refer the reader to recent review in [35]. An example of a traditional, but yet very popular, algorithm for the kind of text mining that could fit our purpose here is Term Frequency-Inverse Document Frequency (TF-IDF) [22]. The idea behind TF-IDF is that a numerical statistic is calculated to reflect how important a word is in a corpus. Its value increases proportionally to the number of occurrences of a word in the document, but is often offsetted by the frequency of the word in the corpus. The purpose of this is to adjust for the fact that some words appear very frequently in general. We refer the reader to the reference in [31] for details on how TF-IDF works.

Examples of machine learning classifiers along with their references are Decision Trees [30], Random Forests [8], K-Nearest Neighbor Classification [12], Support Vector Machines [34]), Logistic Regression [24], Naive Bayes [28], and Gradient Boosting Machine (GBM) classifier [9]. We also refer the user to recent similar applications of data mining and classification in [3, 5]. Figure 2 illustrates that whole methodology of action recommendation for a given event-condition.

3.2 Validating the Learned Actions and Updating the Historical Data

Machine learning classifiers, like the ones we proposed to use in the previous subsection, typically output a score of confidence in the classification assignment. Assuming that the user of our system determines a threshold for the minimum accepted confidence (e.g., 50%), we program the following in our system: If the threshold of assigning an action to an event-condition is equal or above that threshold, then we just apply it. If it is lower, then we let an expert validate that recommended actions of low confidence. This would typically happen when the model encounters new classes of events and conditions pairs.

In the latter case, after the expert review, the expert would either confirm the recommended action or they would update/change it. Note that such expert

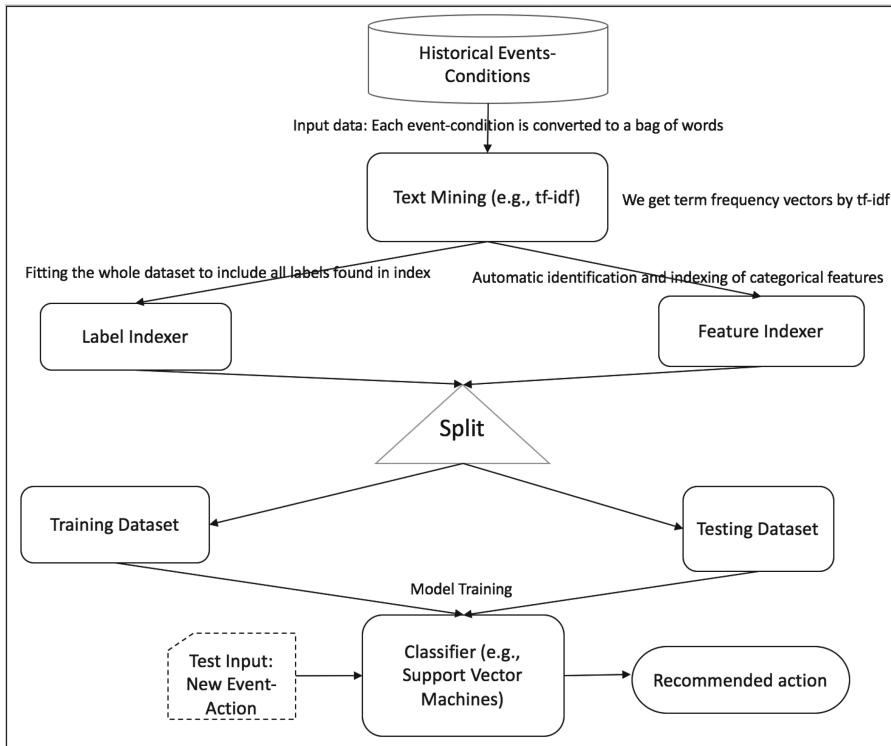


Fig. 2. Illustration of our action recommendation methodology

validation is minimal compared to ad-hoc prior systems where the expert defines all the policies manually. Lastly, the expert recommend actions are used to update the historical events/conditions/actions data to improve the learning model in the future recommendations.

3.3 Learning the Priority of Events-Conditions

The objective of this step is to prioritize the actions recommended by the recommender discussed in Sect. 3.1. We need to do such prioritization because there are typically limited resources in the systems and thus, we might need to do a subset of the recommended actions. Therefore, we want to prioritize these actions so that the ones that the system finally chooses to perform have the highest possible collective priority.

We assume that we are given sets of previous actions and their importance. For example, an action of calling the fire department when a fire occurs is obviously more important than the action of turning off the air conditioning in a building given a certain temperature. That importance could either be a score/weight of importance of each action or a binning classification

(e.g., classifying each action in one of five bins, where the first bin corresponds to most important actions, the second one is less important, ..., etc.).

Now, the way our prioritization works is similar to our recommendation. That is, we perform some text mining on the actions and the classify them into the importance classes. The classification could be a multi-label classification in case the historical data are labeled into bins, or it could be a binary classification in case the prioritization is in the form of a weight for each action. In the latter case, the score that the model gives for each action corresponds to the predicted weight of such action.

After we prioritized the actions, we note that we cannot just simply rank them in order of the predicted priority and keep fulfilling the capacity of the resources that we have with the ones with the highest weight until that capacity is fulfilled. We cannot do that because, first the resources might be multi-dimensional (e.g., CPU, RAM, network bandwidth, and storage of the devices used for the data flow in our IoT system), and secondly, even if it were one-dimensional (i.e., just one resource type), using such greedy algorithm might not make an optimal use of the available resource capacity. Thus, we formulate a mathematical optimization model in the next section in order to optimally choose the final set of actions that can be feasibly performed.

3.4 An Optimization Module for the Final Action Recommendations

Let the set of resources be R and the set of recommended actions be A . We denote the utilization of each action $a \in A$ from each resource $r \in R$ by u_{ar} , and the capacity of resource $r \in R$ by c_r . We also denote the priority/weight of action $a \in A$ by w_a . Lastly, our only set of variables here are the ones related with whether we will choose action $a \in A$ to be executed or not. We let X_a be such variables, where X_a is 1 if action $a \in A$ is chosen, and 0 otherwise. Now, we formulate our mathematical optimization model as follows:

$$\max \quad \sum_{a \in A} w_a \cdot X_a \quad (1)$$

$$\text{s.t.} \quad \sum_{a \in A} u_{ar} \cdot X_a \leq c_r, \quad \forall r \in R \quad (2)$$

$$X_r \in \{0, 1\} \quad (3)$$

Where objective function 1 maximizes the sum of the weights of the actions that are to be chosen. Note that such weight is the score of the action, in case the output of the priority learning step is the weight, and it is the reciprocal of the priority/bin case the given priority is the corresponding bin. Constraint 2 ensures that the total utilization of all chosen actions from each resource is less than or equal to the total capacity of that resource. Lastly, constraint 3 ensures that each of our variables is either zero or one.

Model (1–3) is an integer programming model, which might not be easy to solve. However, we note that its structure is a multidimensional knapsack

problem, which is well studied in the operations research literature. We refer the reader to the review article in [16] for efficient ways of solving such model.

4 Conclusions and Directions for Future Work

In this paper, we proposed a cognitive system for action recommendation action recommendation for policy-based data management in IoT systems. That is, whenever an event-condition happens in an IoT system, we showed how our system can be used to recommend the optimal set of actions to execute, given the resource constraints of the system, and with minimum human intervention.

There are several directions for future work. First, an implementation of our system with a real-world IoT use case would show the effectiveness of our approach. Second, investigation of different machine learning algorithms in the recommendation and priority determination steps would be helpful in determining which ones are most appropriate for our proposed system. Lastly, one direction for future research to our work is incorporating uncertainty in the observed events and conditions, and designing a system that would recommend actions before such certainty is realized.

References

1. Abu-Elkheir, M., Hayajneh, M., Ali, N.A.: Data management for the Internet of Things: design primitives and solution. *Sensors* **13**(11), 15582–15612 (2013)
2. Aceto, G., Botta, A., de Donato, W., Pescapé, A.: Cloud monitoring: a survey. *Comput. Netw.* **57**(9), 2093–2115 (2013)
3. Asthana, S., Megahed, A., Becker, V., Nakamura, T., Gajananan, K.: A cognitive prioritization for reports generated in resource constrained applications. In: 2017 IEEE International Conference on Services Computing (SCC), pp. 418–425. IEEE (2017)
4. Asthana, S., Megahed, A., Strong, R.: A recommendation system for proactive health monitoring using IoT and wearable technologies. In: 2017 IEEE International Conference on Artificial Intelligence and Mobile Services (AIMS), pp. 14–21. IEEE (2017)
5. Asthana, S., Strong, R., Megahed, A.: Healthadvisor: recommendation system for wearable technologies enabling proactive health monitoring. arXiv preprint [arXiv:1612.00800](https://arxiv.org/abs/1612.00800) (2016)
6. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2007)
7. Brandt, J., Gentile, A., Mayo, J., Pebay, P., Roe, D., Thompson, D., Wong, M.: Resource monitoring and management with OVIS to enable HPC in cloud computing environments. In: IEEE International Symposium on Parallel Distributed Processing, May 2009
8. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
9. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
10. Cheng, Y., Chen, W., Wang, Z., Yu, X.: Performance-monitoring-based traffic-aware virtual machine deployment on NUMA systems. *IEEE Syst. J.* **PP**(99) (2015)

11. Clayman, S., Galis, A., Mamatas, L.: Monitoring virtual networks with Lattice. In: Network Operations and Management Symposium Workshops, April 2010
12. Cover, T., Hart, P.: Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**(1), 21–27 (1967)
13. Cruz Huacarpuma, R., de Sousa Junior, R.T., de Holanda, M.T., de Oliveira Albuquerque, R., García Villalba, L.J., Kim, T.H.: Distributed data service for data management in internet of things middleware. *Sensors* **17**(5), 977 (2017)
14. Doraisamy, M., bin Ibrahim, S., Mahrin, M.N.: Metric based software project performance monitoring model. In: Proceedings of the IEEE International Conference on Open Systems (ICOS). IEEE, August 2015
15. Enterprises, N.: Nagios Documentation (2014). <http://www.nagios.org/documentation>
16. Fréville, A.: The multidimensional 0–1 knapsack problem: an overview. *Eur. J. Oper. Res.* **155**(1), 1–21 (2004)
17. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. SSS. Springer, New York (2009). <https://doi.org/10.1007/978-0-387-84858-7>
18. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013)
19. Jeswani, D., Natu, M., Ghosh, R.K.: Adaptive monitoring: a framework to adapt passive monitoring using probing. In: Proceedings of the 8th International Conference on Network and Service Management, CNSM 2012, pp. 350–356. International Federation for Information Processing, Laxenburg, Austria (2013)
20. Jeswani, D., Natu, M., Ghosh, R.K.: Adaptive monitoring: application of probing to adapt passive monitoring. *J. Netw. Syst. Manag.* **23**(4), 950–977 (2015)
21. Katsaros, G., Gallizo, G., Kübert, R., Wang, T., Fitó, J.O., Henriksson, D.: A Multi-level architecture for collecting and managing monitoring information in cloud environments. In: Leymann, F., Ivanov, I.I., van Sinderen, M., Shishkov, B. (eds.) Proceedings of the Third International Conference on Cloud Computing and Services Science, CLOSER. SciTePress (2011)
22. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2014)
23. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Comput.* **30**(7), 817–840 (2004)
24. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers. Wiley, Hoboken (2010)
25. Munawar, M.A., Reidemeister, T., Jiang, M., George, A., Ward, P.A.S.: Adaptive monitoring with dynamic differential tracing-based diagnosis. In: De Turck, F., Kellerer, W., Kormentzas, G. (eds.) DSOM 2008. LNCS, vol. 5273, pp. 162–175. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87353-2_13
26. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)
27. Natu, M., Sethi, A.S.: Probabilistic fault diagnosis using adaptive probing. In: Clemm, A., Granville, L.Z., Stadler, R. (eds.) DSOM 2007. LNCS, vol. 4785, pp. 38–49. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75694-1_4
28. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In: Advances in Neural Information Processing Systems 2, pp. 841–848 (2002)
29. Padiya, T., Bhise, M., Rajkotiya, P.: Data management for Internet of Things. In: Region 10 Symposium (TENSYMP), pp. 62–65. IEEE (2015)

30. Quinlan, J.R.: C4.5: Programs for Machine Learning. Elsevier, New York (2014)
31. Ramos, J., et al.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, pp. 133–142 (2003)
32. Rivera, J., van der Meulen”, R.: Gartner says 4.9 billion connected ‘things’ will be in use in 2015 (2014). <http://www.gartner.com/newsroom/id/2905717>
33. Roman, R., Najera, P., Lopez, J.: Securing the internet of things. Computer **44**(9), 51–58 (2011)
34. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press, Cambridge (2002)
35. Sharma, K., Sharma, A., Joshi, D., Vyas, N., Bapna, A.: A review of text mining techniques and applications. Int. J. Comput. (IJC) **24**(1), 170–176 (2017)
36. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in Internet of Things: the road ahead. Comput. Netw. **76**, 146–164 (2015)
37. Tata, S., Megahed, A., Mohamed, M., Nazeem, A., El Harouni, A.: Data collection and management in IoT environments: challenges and future directions. In: 2017 IEEE International Congress on Big Data (BigData Congress). IEEE (2017)
38. Tata, S., Mohamed, M., Megahed, A.: An optimization approach for adaptive monitoring in IoT environments. In: 2017 IEEE International Conference on Services Computing (SCC), pp. 378–385. IEEE (2017)
39. Trihinas, D., Pallis, G., Dikaiakos, M.D.: AdaM: An adaptive monitoring framework for sampling and filtering on IoT devices. In: IEEE International Conference on Big Data. IEEE Computer Society, Los Alamitos, CA, USA (2015)
40. Zarko, I.P., Pripuzic, K., Serrano, M., Hauswirth, M.: IoT data management methods and optimisation algorithms for mobile publish/subscribe services in cloud environments. In: 2014 European Conference on Networks and Communications (EuCNC), pp. 1–5. IEEE (2014)



A Research Perspective on Fog Computing

David Bermbach¹(✉), Frank Pallas¹, David García Pérez², Pierluigi Plebani³,
Maya Anderson⁴, Ronen Kat⁴, and Stefan Tai¹

¹ TU Berlin, Information Systems Engineering Research Group, Berlin, Germany
{db,fp,st}@ise.tu-berlin.de

² Atos Spain SA, Atos Research and Innovation, Barcelona, Spain
david.garciaperez@atos.net

³ Politecnico di Milano, Milan, Italy
pierluigi.plebani@polimi.it

⁴ IBM Research Haifa, Haifa, Israel
{mayaa,ronenkat}@il.ibm.com

Abstract. State-of-the-art applications are typically deployed on top of cloud services which offer the illusion of infinite resources, elastic scalability, and a simple pay-per-use billing model. While this is very convenient for developers, it also comes with relatively high access latency for end users. Future application domains such as the Internet of Things, autonomous driving, or future 5G mobile apps, however, require low latency access which is typically achieved by moving computation towards the edge of the network. This natural extension of the cloud towards the edge is typically referred to as Fog Computing and has lately found a lot of attention. However, Fog Computing as a deployment platform has not yet found widespread adoption; this, we believe, could be helped through a consistent use of the service-oriented computing paradigm for fog infrastructure services. Based on this motivation, this paper describes the concept of Fog Computing in detail, discusses the main obstacles for Fog Computing adoption, and derives open research challenges.

Keywords: Fog Computing · Cloud computing · Edge computing

1 Introduction

Today's state-of-the-art applications are typically deployed on top of cloud services, thus, leveraging cost benefits, ease-of-use, elastic scalability, and the illusion of infinite resources [12]. While cloud services come with these obvious benefits, they also have a major disadvantage: cloud data centers are centralized and, thus, typically far from the end user resulting in high access latencies. This is sufficient for many application domains such as enterprise or web applications but not for more modern application domains such as autonomous driving, Internet of Things (IoT)-based platforms, or 5G mobile applications. Therefore, these

applications are typically deployed on edge devices. However, while such edge devices offer low access latencies due to their physical proximity to end users, they also come with limited resources and are subject to availability problems.

In this situation, an obvious approach is to use both cloud services and edge nodes at the same time to achieve low latency while having access to scalable, infinite resources. This paradigm, which has recently emerged as a natural extension of Cloud Computing, is typically referred to as Fog Computing [4,13]. Beyond the benefits already discussed, Fog Computing also promises better privacy to end users: While current, cloud applications collect personal data need in a central place, future Fog applications can keep detailed personal data at the edge, transferring only aggregated or properly anonymized data to the cloud.

These obvious advantages of Fog Computing and the public attention it created notwithstanding, there has so far been surprisingly little adoption of the paradigm for actual applications. We believe that there are several reasons for this, chief among which we see a lack of edge service offerings: managing edge nodes manually is a gruesome task and requires massive upfront investments – the exact opposite of cloud services with their convenience focus and pay-as-you-go model. Also, it is still not completely clear what Fog Computing actually is.

This paper aims to shed some light on Fog Computing and its role in service-oriented computing (SOC). To this aim, we firstly provide a definition of Fog Computing and the involved concepts relevant to SOC. Secondly, we identify the main obstacles for widespread fog adoption and describe a number of fundamental research challenges. For this reason, this paper should, thus, be seen as a call to action and as an overview of challenges that we as researchers should tackle.

This paper is structured as follows: In Sect. 2, we give an overview of state-of-the-art Fog Computing (along with its competing definitions as well as its benefits and opportunities). Afterwards, in Sect. 3, we identify and discuss obstacles for widespread fog adoption. In Sect. 4, we describe research challenges that result from these obstacles before concluding.

2 From Cloud to Fog Computing

Fog computing has been initially introduced in the telecommunication sector [4] when researchers and practitioners realized how the role of the final users changed from *consumers* of information to *prosumers* (producers and consumers at the same time). In fact, the original paradigm on which the Web is based assumes that the core of the network is in charge of providing information that will be consumed at the edge. Prosumers with mobile devices or IoT sensors, however, generate immense data quantities at the edge of the network.

So, what precisely is Fog Computing and how can it be distinguished from Edge Computing? Edge Computing is exclusively about computation at the edge of the network without any notion of cloud services. Depending on the source, Fog Computing is either the same as Edge Computing [20] or is defined as the amalgam of cloud, edge, and any intermediary nodes in between (this could be

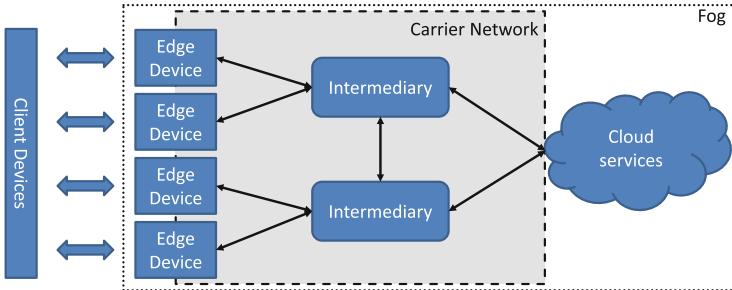


Fig. 1. Deployment overview of Fog Computing

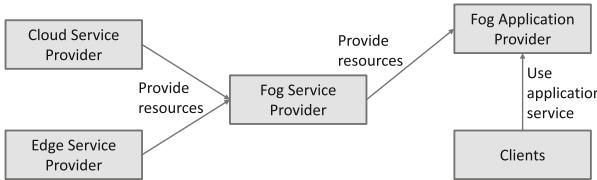


Fig. 2. Overview of roles in Fog Computing

small- to medium-sized data centers within the core network of the network provider) [21]. In this paper, we take the latter perspective. See also Fig. 1 for a high-level deployment overview of Fog Computing.

To realize its full potential, Fog Computing must be more than creating a *data center in the box*, i.e., Cloudlets [19], to bring the cloud closer to data producers. Instead, Fog Computing must be seen as a “resource layer that fits between the edge devices and the cloud data centers, with features that may resemble either.” [25]. As also pointed out by the OpenFog Consortium [13], the goal of Fog Computing is to provide a set of methods and tools to create a continuum between edge and cloud. For this, Fog Computing technologies especially need to enable fluid data movement between cloud services and edge, in both directions, while satisfying application constraints in terms of quality of service (QoS). Such data movement may also be accompanied by movement of computation – both in the same way and in a complimentary fashion compared to data movement.

With the advent of Fog Computing, applications based on this paradigm can exploit the advantages of both the edge and cloud environments. With specifically tailored frameworks, e.g., [17], developers can leave it to the Fog Computing layer to automatically handle data and computation placement. Based on such dynamic data and computation movement, applications can provide new functionality or new QoS levels (e.g., significantly lower latency) and use novel ways for dynamically balancing tradeoffs. The lack of such capabilities has so far hindered the emergence of technologies such as autonomous and interconnected driving, large-scale IoT, etc., which should significantly profit from fog adoption.

While Cloud Computing is largely centered around a service concept, this is not (yet) the case for the edge as we will also discuss in the next sections. However, we firmly believe that the SOC paradigm might prove valuable for widespread fog adoption. In this context, we envision the following roles (see also Fig. 2): Already today, cloud service providers offer access to cloud infrastructure services. Edge service providers may do the same for edge resources. Fog service providers, in contrast, offer service-based access to fog infrastructure resources – in this capacity, fog service providers may act as resellers. Alternatively, two or more of these roles may be held by the same organization. Finally, Fog application providers rent resources through fog infrastructure services to offer their application to clients. We explicitly use the broad term “client” as this may include IoT devices, mobile phones, or any other end user.

However, there are several obstacles and yet unsolved research challenges that need to be faced in order to actually pave the way for a broad adoption of Fog Computing. In literature, some other papers already made the attempt to identify these challenges. For instance, even if it does not mention Fog Computing, Díaz et al. [6] analyses the challenges when considering the integration of IoT and cloud services, thus, in a scenario close to Fog Computing. Yet, [5] proposes a Fog Computing definition close to ours and also identifies some possible applications as well as research challenges mainly concerning the infrastructural level, while in this paper we are taking a broader perspective also considering the application level. Finally, in [20], even if mainly focused on the Edge Computing, a list of challenges are introduced with more emphasis on the programming models.

As already mentioned, the goal of the paper is to consider Fog Computing as an environment in which both cloud and edge resources are used as application service deployment environments. Obstacles to the adoption of Fog Computing, and the related research challenges arising when trying to overcome to those obstacles, necessarily include the design, deployment, and management of complex and distributed systems, as discussed in the following sections.

3 The Main Obstacles for Adoption of Fog Computing

Broadly, obstacles for a wide adoption of Fog Computing can be categorized as *inherent*, e.g., available technology or physical constraints, and *external* obstacles, e.g., privacy legislation. In this section, we will give an overview of both.

3.1 Inherent Obstacles

As inherent obstacles we see those that result from the very idea of using fog resources. Specifically, these can be technical constraints such as a given limit of computational power, logical constraints such as having tradeoffs in distributed systems, or simply market constraints such as the fact that there are currently no managed edge services.

O1: No Edge Services. While the cloud is conveniently usable with its service-based consumption model, there are currently no edge infrastructure services where compute or storage capacity could be provisioned on-demand. We believe that managed edge services are bound to enter the market sooner or later; possibly even as fog services in cooperation of cloud providers and network carriers. Currently, Amazon is taking first steps in that direction through their Green-grass “service”¹ which provides software for edge devices. However, since there is currently no way to really “rent” on-demand edge capacity, this means that fog application providers currently need to build, manage, and run their own physical edge “boxes” including cloud integration.

O2: Lack of Standardized Hardware. While there are some ready-to-use, off-the-shelf edge “boxes”, these come in a variety of flavors, e.g., regarding compute power. Alternatively, Raspberry Pis, BeagleBoards, or custom solutions can be used. This leads to a broad heterogeneity of edge node hardware which has two effects: First, software stacks need to be adapted or may not even run everywhere. Systems that can actually run everywhere are unlikely to fully tap the potential of the respective hardware resources. Second, application architectures need to be able to deal with such diverse resource capacities. This requires highly modularized applications that can deliver some or all service features depending on the nodes where they are running.

O3: Management Effort. Since Fog Computing is mainly about bringing data and computation closer to end users, dense fog deployments will result in very high numbers of fog nodes. These all need to be managed, e.g., when scaling, installing updates, or updating configurations. As there is currently no managed fog infrastructure service, this management effort is left with application providers. In comparison small on-premise data centers, this requires immense effort.

O4: Managing QoS. Complexity of systems typically increases with the number of nodes in the system and their geographical distribution. This is due to issues and faults – ranging from simple ones such as network latencies, message reordering, or message loss to complicated faults such as network partitioning or byzantine failures. Usually, systems can cope with these issues but they still affect QoS and their tradeoffs. In large-scale geo-distributed systems, all these issues become more pronounced and faults happen more often – simply based on probabilities and the increased number of distributed nodes.

At the same time, fog application domains such as IoT or autonomous driving actually have *stronger* quality requirements on infrastructure services as the potential impact of “things going wrong” is much more severe and affects the physical world. E.g., faults in a social network may lose a private message;

¹ aws.amazon.com/greengrass.

identical faults could result in injuries and deaths if they affect safety-critical functionality of autonomous cars.

O5: No Network-Transparency. Distributed systems research has always tried to hide distribution, i.e., that applications will run as if they were running on a single machine. Over the years, more and more of these transparencies needed to be sacrificed in favor of other QoS goals. What has still remained hidden so far is the network layout; e.g., cloud services expose some basic high-level information on their setup and distribution (e.g., availability zones and regions in AWS) and heavily rely on network virtualization. However, in Fog Computing, the interconnection of nodes on a logical level can no longer be done in arbitrary ways as the underlying networks are, in fact, more or less hierarchical; e.g., two geographically near edge nodes may be connected directly, on the next higher hierarchy level, at the Internet backbone level or anywhere in between. In geo-distribution, this results in edge-to-edge latencies that are magnitudes apart. Therefore, fog application services can no longer be based on high-level abstractions describing distance notions (e.g., an AWS region). Instead, they need to be aware of actual network topologies unless they can cope with unexplainable performance variance across nodes. For manageability reasons, this calls for the introduction of novel, more fine-grained topology abstractions that can be handled inside application services.

3.2 External Obstacles

Beyond the inherent obstacles already discussed, there are also influence factors resulting from external entities such as government agencies or attackers.

O6: Physical Security. Traditional data centers can employ arbitrarily complex measures for physical access control including but not limited to onsite security staff. Typically, this is subject to audit and certification to assert that their physical security measures are up to par. For fog nodes widely distributed across a basically hostile environment, this is impossible. Here, physical security may mean to attach an edge box to the top of a street light's pole (instead of at eye level) or surrounding it with a fire-resistant coating to counter vandalism. What is less obvious, though, is that potential attackers gain a physical attack vector into a software system which enables an entire set of so far theoretic attacks. Even though there are existing approaches to comparable problems from other fields – e.g., for tamper-proofing smart energy meters or for protecting modern cars against software manipulations – these are typically addressing highly specific risks that are well-known in advance. Also, such approaches have only been used in specialized embedded systems yet. They can, thus, not be directly applied to heterogeneous, multi-purpose fog nodes. Nonetheless, they might provide valuable inspiration and direction for addressing physical security in Fog Computing; e.g., one could design fog nodes so that they can detect tampering to then shut down the node, move and wipe all data, and notify the authorities. Application services, in turn, need to be able to cope with such behavior.

O7: Legal and Regulatory Requirements. In many application domains, e.g., in health care, there are legal and regulatory requirements that mandate data to be held in certain physical locations. Also, storage and processing facilities must demonstrably meet certain requirements. While cloud data centers can be certified if they exist in the respective region, this seems impractical for edge nodes. Furthermore, data privacy regulation such as the EU’s GDPR [9] also include aspects like transparency, i.e., the data subjects’ right to know where their personal data is stored. In liquid fog-based applications, this is challenging to prove as the application may not even know the data location due to virtualization.

4 Open Research Challenges in Fog Computing

While we as a research community cannot start a managed edge infrastructure service, we can support fog adoption through research efforts on the research challenges presented in this section. Even though there are interdependencies between these challenges, they can be broadly categorized regarding the role that they mainly affect: fog infrastructure service providers or fog application providers. In a third group are management challenges such as failure handling or security and privacy aspects which fog application providers are primarily but not exclusively responsible for. In this section, we will present this non-conclusive list of main challenges following the above-mentioned ordering structure; where possible we also point out possible avenues for addressing them.

4.1 Fog Service Providers

While our research is unlikely to directly result in fog infrastructure services, there are still a number of open research challenges that potential fog service providers need to face once they decide to enter the market.

RC1: New Abstractions. One of the main goals of using a SOC model is to hide implementation or runtime details. In the case of fog resources, geo-distribution, physical locations, and sometimes the node type actually matter. Still, fog application providers seem ill equipped to deal with services that expose individual edge locations and capacities. What we need is some middle ground that exposes enough details on distribution and physical locations so that applications can work with it but are not overwhelmed by it. We believe that the event-driven programming model of typical (serverless) Function as a Service (FaaS) [18] offerings naturally lends itself to this: Applications can use policies to define properties on various levels, e.g., for the case of location, based on density for a given region, everywhere within a given region, or based on concrete coordinates, which are then used by the fog FaaS offering to deploy functions accordingly. Of course, other programming models are also an option – however, a serverless approach seems like an easy-to-use abstraction that allows control over deployments while not forcing unnecessary details on application developers.

RC2: Capacity Management. One of the main properties of cloud services is the illusion of infinite resources [12]; for edge nodes, in contrast, upholding this illusion becomes impossible. Essentially, in an edge node there is a limited set of resources that is available to fog service consumers. In the cloud, resource pooling across a large customer base and inside gigantic data centers helps cloud providers even out natural demand variability. In Edge Computing, the limited capacity of nodes does not allow this: edge nodes will alternate between states of idleness and periods with more demand than they can handle. However, offloading processing and data to adjacent nodes or intermediary nodes is bound to impact QoS. This leads to the following three main questions for fog service providers:

1. How shall scarce capacity be distributed across applications in periods of high demand?
2. Shall fog service providers deny requests that cannot be fulfilled or shall they fulfill them partially, e.g., on other nodes, resulting in poorer QoS behavior?
3. How shall fog service providers size their edge and intermediary nodes if demand is unknown before deployment?

We believe that auctioning-based approaches (comparable to virtual machine spot markets [14]) can help us tackle the first two questions. E.g., an application could submit different price bids for specific groups of nodes (e.g., a specific edge node compared to an intermediary node with slightly higher latencies). Whenever demand exceeds capacity, this allows fog service providers to auction off free capacity to the highest paying bidder. If bids reflect individual utilities, this results in an (economically) optimal distribution of capacity where optimizations can be decided locally. Of course, our community could develop more complex mechanisms, e.g., distributions based on QoS requirements etc.

Regarding the third question, we believe that fog service providers should strive to design modular edge nodes where additional capacity can be added or removed incrementally, e.g., a rack of Raspberry Pis.

4.2 Fog Application Providers

There is a number of key differences between cloud-native and fog-native applications. Beyond dealing with abstractions offered by (potential) fog service providers, application services need to be able to run in a widely distributed setup (resulting in new modularization challenges), must be able to dynamically adapt to new environments (meaning a unique “fluidity” challenge), and must consider – on an architectural level – how to integrate various fog nodes ranging from high performance cloud machines to Raspberry Pi-based edge nodes.

RC3: Modularization. State-of-the-art modularization of applications typically follows a microservice-based approach [11] where each microservice represents a vertical cut through the application stack. This means that a microservice will usually have its own storage layer and may even use its own technology

stack. This modularization is largely driven by organizational decisions, i.e., a company organized into small DevOps teams will assign part of the functionality to each of the teams who will then be responsible for developing and running that microservice. Service-oriented architectures, in contrast, were mainly driven by technical design goals such as reusability, less by organizational factors. Now, in future fog applications, it may make sense to still follow a microservice-based approach (or it may not – depending on the organizational structure of the developer group), but within a microservice not all parts of the stack may run on all nodes, also considering that replication in the deployment suddenly becomes widely distributed. What we envision is that the vertical cuts of microservices are cut horizontally into smaller slices that can be distributed. For instance, an application service may decide to keep some basic presentation and application logic plus a caching layer at the edge and persistently store data on cloud nodes. For this, we need further research on how to slice microservices into even smaller pieces that can be replicated, shuffled around, and distributed widely.

RC4: Fluidity. Novel fog applications will have to be designed in a way that openly embraces a notion of fluidity: application modules will be interrupted frequently, will move between fog nodes, and maybe cloned at any time for such reasons as limited edge capacity, moving client devices, cost optimizations, or changing QoS requirements. Part of fluidity is also the ability to deal with varying resource capacity in nodes, e.g., when moving from a high performance cloud node to a small edge node. A convenient way to address this fluidity challenge is to strictly separate stateful and stateless components – an approach also (originally) taken with containers such as Docker [7]. The same paradigm can also be found in serverless FaaS deployments where applications are broken down into short-lived, stateless tasks that interact with a stateful storage tier. Research should focus on application designs that can cope with such degrees of fluidity.

4.3 Cross-Cutting Concerns

Failure handling primarily falls into the domain of application developers; however, depending on the actions of prospective fog service providers, failure handling activities can be severely affected. E.g., applications that use bare resources will have to employ fundamentally different mechanisms than applications that run on top a fog-based FaaS offering. Finally, as in Cloud Computing [10], security and privacy aspects are obvious candidates for further research challenges.

RC5: Graceful Degradation and Fault-Tolerance. Fault-tolerance is already challenging in the cloud – fog services with more nodes, wider distribution, and more complex and dynamic interdependencies between components make this even harder. At the same time, fog application domains such as autonomous driving or IoT tend to have more stringent QoS requirements as unhandled failures are bound to have physical effects. Based on this, it is

of utmost importance to handle and hide failures where possible and to support graceful degradation, e.g., in the form of a “safe mode”, when not. How this can be achieved in fog applications or infrastructure services is still unclear. Suitable safeguards that monitor necessary requirements and notify in case of violations might be alternatives as well as compensation and data interpolation mechanisms. E.g., in autonomous driving, a fog storage service should aim to underestimate distances between cars when forced to in the presence of missing data as an overestimating interpolation may result in accidents. Here, research should work on identifying failure detectors or use case-driven compensation mechanisms.

RC6: Benchmarking and Testing. Realistic benchmarking [3] and testing are already challenging for cloud services: Both can easily be executed under controlled conditions (much work in benchmarking is dedicated to creating controlled environments) whereas interesting behavior, e.g., [1, 2, 16] may only be observable in realistic conditions, i.e., at scale and in a regular deployment while running a production workload. For fog-based applications, additional influence factors like heterogeneous hardware (potentially under control by different legal entities), wide geo-distribution, and only occasionally (re)occurring failure situations (possibly resulting from unforeseeable events in the “physical world”) are added to the melee of challenges. In short, it becomes completely impossible to accurately emulate a production system for testing purposes since the cost of running such a setup is prohibitive and a second set of comparable resources may not even exist. Future research in this area should focus more on continuous benchmarking [24] and testing where both are executed during a build process: starting from a small isolated test setup to test basic QoS and functionality, gradually rolling out the software to the target environment. In short, the differences between benchmarking and testing on the one side and monitoring on the other side will gradually be blurring. Research should try to identify how this can be done in practice without impairing production systems. Furthermore, more research needs to be directed towards benchmarking fault-tolerance and related qualities.

RC7: Attack-Agnostic Services. Fog Computing also raises a number of security challenges, mainly how to provide appropriate security in an essentially hostile environment. Besides technical mechanisms, this might also require novel views to the weighing between acceptable risks and the costs of countermeasures. Especially with regard to the physical attack vectors omnipresent in the context of edge nodes, it seems highly worthwhile to put more effort on *coping* with attacks instead of *avoiding* them. We, thus, see the design of “attack-agnostic” services that accept certain attacks as unpreventable and implement measures for detecting and reacting to them (particularly to safeguard data) as a promising strand of research. Integrating such approaches into established security approaches for distributed service systems is another research question. Finally, the limited capabilities of edge nodes may also restrict the applicability

of security mechanisms widely adopted today: Encryption overheads – all too often already mistaken as virtually nonexistent [15] – may suddenly become restricting factors in the context of constrained edge nodes and. Also, the use of well-established concepts for access management and enforcement may prove inappropriate due to the common unreachability of centralized components, calling for novel schemes tailored to the specific givens in fog scenarios [23].

RC8: Compliance with Privacy Legislation. For privacy, Fog Computing provides both opportunities and challenges. Fulfilling data privacy requirements that give data subjects the right to know and to control where their personal data resides is particularly challenging in such a widely distributed, ever-changing network. Other regulations, e.g., regarding auditability or the physical placement of data are also difficult to address – possibly through information flow control, e.g., [22], if it can be implemented with little overhead.

On the other hand, Fog Computing also provides a number of opportunities, e.g., data could be aggregated or anonymized on edge nodes before forwarding it to a “central” cloud service, i.e., sensitive, detailed data may not even leave the direct vicinity of the data subject [8]. Similarly, the processing of sensitive personal data may be relocated away from centralized data lakes to the place of data collection, serving privacy goals like data minimization and purpose limitation.

5 Conclusion

While Cloud Computing can today be considered well established, modern application domains such as IoT, autonomous driving, or even mobile applications trying to tap the full potential of future 5G networks require an extension of the cloud towards the edge, thus, naturally leading to the new Fog Computing paradigm. In Fog Computing, application services run on both edge nodes (with low latency access but very limited resource capabilities) and in the cloud (with higher access latency but practically unlimited resources) as well as on possible intermediary nodes. Fog Computing as a new paradigm is a yet virtually unexplored field that offers a number of open research challenges.

In this paper, we started by describing what Fog Computing actually is – also in comparison to Edge Computing and Cloud Computing. Afterwards, we described the main obstacles for widespread fog adoption, distinguishing inherent and external challenges. An example for inherent challenges is the current lack of an edge service model which leads to a “build your own infrastructure” trend with all its undesirable implications; examples for external challenges are physical security or regulations. Building on these identified obstacles, we then described open research questions that arise for either (potential) fog service providers or fog applications running on top of such services. We also identified a number of cross-cutting concerns which fall into the domain of fog service consumers and application providers but are strongly affected by fog service provider actions, e.g., fault-tolerance, security, privacy, but also benchmarking and testing.

All in all, this paper aims to identify a research agenda in Fog Computing where we, as service-oriented computing researchers, as well as other research communities can contribute. Where feasible, we also pointed out specific directions that we believe might prove useful for overcoming the named obstacles or for addressing the resulting research questions.

Acknowledgments. This work has been supported by the European Commission through the Horizon 2020 Research and Innovation program under contract 731945 (DITAS project).

References

1. Bermbach, D., Tai, S.: Benchmarking eventual consistency: Lessons learned from long-term experimental studies. In: Proceedings of IC2E. IEEE (2014)
2. Bermbach, D., Wittern, E.: Benchmarking web API quality. In: Bozzon, A., Cudré-Maroux, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 188–206. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_11
3. Bermbach, D., Wittern, E., Tai, S.: Cloud Service Benchmarking. Measuring Quality of Cloud Services from a Client Perspective. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55483-9_1
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of MCC. ACM (2012)
5. Dastjerdi, A.V., Buyya, R.: Fog computing: helping the internet of things realize its potential. Computer **49**(8), 112–116 (2016)
6. Díaz, M., Martín, C., Rubio, B.: State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. J. Netw. Comput. Appl. **67**, 99–117 (2016)
7. Ernst, D., Bermbach, D., Tai, S.: Understanding the container ecosystem: A taxonomy of building blocks for container lifecycle and cluster management. In: Proceedings of WoC. IEEE (2016)
8. Esposito, C., Castiglione, A., Pop, F., Choo, K.K.R.: Challenges of connecting edge and cloud computing: a security and forensic perspective. IEEE Cloud Comput. **4**(2), 13–17 (2017)
9. European Parliament and European Council: Regulation 2016/679 - general data protection regulation - GDPR (2016)
10. Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: Above the clouds: A berkeley view of cloud computing. University of California, Berkeley, Technical report (2009)
11. Lewis, J., Fowler, M.: Microservices: a definition of this new architectural term. ThoughtWorks (2014). Accessed 1 Jun 2017. <http://martinfowler.com/articles/microservices.html>
12. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Special Publication 800-145 (2011)
13. OpenFog Consortium Architecture Working Group: OpenFog Architecture Overview (2016). Accessed 8 Aug 2017. <http://www.openfogconsortium.org/ra>
14. Ouyang, X., Irwin, D., Shenoy, P.: Spotlight: An information service for the cloud. In: Proceedings of ICDCS. IEEE (2016)
15. Pallas, F., Bermbach, D., Müller, S., Tai, S.: Evidence-based security configurations for cloud datastores. In: Proceedings of SAC. ACM (2017)

16. Pallas, F., Günther, J., Bermbach, D.: Pick your choice in hbase: Security of performance. In: Proceedings of BigData. IEEE (2016)
17. Plebani, P., Garcia-Perez, D., Anderson, M., Bermbach, D., Cappiello, C., Kat, R.I., Pallas, F., Pernici, B., Tai, S., Vitali, M.: Information Logistics and Fog Computing: The DITAS Approach. In: Proceedings of CAISE Forum. CEUR (2017)
18. Roberts, M.: Serverless architectures (2016). Accessed 1 Jun 2017. <http://martinfowler.com/articles/serverless.html>
19. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
20. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
21. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016)
22. Singh, J., Pasquier, T.F.J.M., Bacon, J., Diaconu, R., Powles, J., Evers, D.: Big Ideas paper: Policy-driven middleware for a legally-compliant Internet of Things. In: Proceedings of MIDDLEWARE. ACM (2016)
23. Stojmenovic, I., Wen, S., Huang, X., Luan, H.: An overview of fog computing and its security issues. Concurr. Pract. Exp. **28**(10), 2991–3005 (2016)
24. Tai, S.: Continuous, trustless, and fair: changing priorities in services computing. In: Lazovik, A., Schulte, S. (eds.) ESOCC 2016. CCIS, vol. 707, pp. 205–210. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72125-5_16
25. Varshney, P., Simmhan, Y.: Demystifying fog computing: Characterizing architectures, applications and abstractions. CoRR (2017). <http://arxiv.org/abs/1702.06331>

Workshop on Engineering Service-Oriented Applications and Cloud Services

Introduction to the 13th International Workshop on Engineering Service-Oriented Applications and Cloud Services (WESOACS'17)

Andreas S. Andreou¹, Luciano Baresi², George Feuerlicht³,
Winfried Lamersdorf⁴, Guadalupe Ortiz⁵, and Christian Zirpins⁶

¹ Cyprus University of Technology

andreas.andreou@cut.ac.cy

² Politecnico di Milano

luciano.baresi@polimi.it

³University of Economics, Prague

george.feuerlicht@vse.cz

⁴University of Hamburg

lamersdorf@informatik.uni-hamburg.de

⁵University of Cádiz

guadalupe.ortiz@uca.es

⁶Karlsruhe University of Applied Sciences

christian.zirpins@hs-karlsruhe.de

The *International Workshop on Engineering Services-Oriented Applications and Cloud Services (WESOACS)* is a long-established forum (formerly known as WESOA) for innovative ideas from research and practice in the field of software engineering for modern service-oriented application systems. This year, the 13th meeting took place on 13th November in Malaga, Spain.

Service-oriented applications play an important role in many areas, such as enterprise computing, cloud computing, and the Web. While there is agreement on the main principles for designing and developing application systems based on distributed software services, methods and tools that support the development of such applications are still the subject of intense research. These research topics include software service life cycle development methodologies, service-oriented enterprise architectures, service-oriented analysis and design, and in particular service engineering technologies for cloud computing environments in general, and more specifically for current trends in cloud-based applications such as intelligent cyber-physical systems.

Currently, there is a shift in this area to so-called “DevOps” approaches of software development in which software service development and operations are continuously and inextricably linked to achieve faster application delivery with automated release and deployment. Agile processes, microservices, continuous delivery, containers and cloud technologies are just some of the popular topics that contribute to the current IT transformation in this context.

The WESOACS technical program included ten research papers in four thematic sessions. The first one grouped recent developments in *architectures* of cloud frameworks (George Feuerlicht, University of Economics, Prague) and for pattern-based control of privacy policies in the cloud (Stefan Schönen, University of Duisburg-Essen). In addition, *case studies* on current workflow management systems (Jörg Lenhard, Karlstad University) and on migration between web service API technologies (Maximilian Vogel, University of Applied Sciences Karlsruhe) were presented.

A focus of this year's contributions was on service-oriented development of *intelligent systems*. This included work on access control of web-based smart home platforms (Sebastian Werner, TU Berlin), collaborative environments for business processes with social interaction (Andrea Delgado, University of Uruguay) and interactive assistance systems for multimodal mobility in smart cities (Christian Kuster, TU Berlin).

In addition, new approaches to the development of software services for *cyber-physical systems* were discussed. Here, participants presented approaches for event processing of sensor data for sustainable waste management (Juan Boubeta-Puig, University of Cadiz), knowledge-based service components in production systems (Christopher Haubbeck, University of Hamburg) and the control of intelligent devices by business process management systems (Robert Wehlitz, University of Leipzig).

In the course of the one-day workshop, the participants had ample opportunity for professional exchange and networking, so that the 13th edition of the event can once again be regarded as a complete success.

Organization

Workshop Organizers

Andreas S. Andreou	Cyprus University of Technology, Cyprus
Luciano Baresi	Politecnico di Milano, Italy
George Feuerlicht	University of Economics, Prague
Winfried Lamersdorf	University of Hamburg, Germany
Guadalupe Ortiz	University of Cadiz, Spain
Christian Zirpins	Karlsruhe University of Applied Sciences, Germany

Program Committee

Juan Boubeta-Puig	University of Cádiz, Spain
Alena Buchalcevova	University of Economics, Prague, Czech Republic
Javier Cubo	University of Malaga, Spain
Andrea Delgado	Universidad de la Repùblica, Uruguay
Alfonso Garcia-de-Prado	University of Cádiz, Spain
Laura González	Universidad de la Repùblica, Uruguay
Sam Guinea	Politecnico di Milano, Italy
Kai Jander	University of Hamburg, Germany
Mark Little	Red Hat, UK
Massimo Mecella	SAPIENZA Università di Roma, Italy
Giovanni Quattrocchi	Politecnico di Milano, Italy
Wolfgang Reisig	Humboldt-University Berlin, Germany
Norbert Ritter	University of Hamburg, Germany
Damina Tamburri	DEIB Politecnico di Milano, Italy
Erik Wittern	IBM T.J. Watson Research Center, USA

Acknowledgements. We wish to thank all authors for their contributions, the program committee members whose expert input made this workshop possible as well as the ICSOC'17 workshop co-chairs Lars Braubach and Juan M. Maurillo. G. Ortiz thanks for support from the MINECO/FEDER Funds through project TIN2015-65845-C3-3-R.



Lessons Learned from Evaluating Workflow Management Systems

Jörg Lenhard¹(✉), Vincenzo Ferme², Simon Harrer³, Matthias Geiger³,
and Cesare Pautasso²

¹ Department of Mathematics and Computer Science,
Karlstad University, Karlstad, Sweden

joerg.lenhard@kau.se

² Software Institute, Faculty of Informatics, USI, Lugano, Switzerland
{vincenzo.ferme,cesare.pautasso}@usi.ch

³ Distributed Systems Group, University of Bamberg, Bamberg, Germany
{simon.harrer,matthias.geiger}@uni-bamberg.de

Abstract. Workflow Management Systems (WfMSs) today act as service composition engines and service-oriented middleware to enable the execution of automated business processes. Automation based on WfMSs promises to enable the model-driven construction of flexible and easily maintainable services with high-performance characteristics. In the past decade, significant effort has been invested into standardizing WfMSs that compose services, with standards such as the Web Services Business Process Execution Language (WS-BPEL) or the Business Process Model and Notation (BPMN). One of the aims of standardization is to enable users of WfMSs to compare different systems and to avoid vendor lock-in. Despite these efforts, there are many expectations concerning portability, performance efficiency, usability, reliability and maintainability of WfMSs that are likely to be unfulfilled. In this work, we synthesize the findings of two research initiatives that deal with WfMSs conformance and performance benchmarking to distill a set of lessons learned and best practices. These findings provide useful advice for practitioners who plan to evaluate and use WfMSs and for WfMS vendors that would like to foster wider adoption of process-centric service composition middleware.

Keywords: Workflow management systems · Standards
Lessons learned · Evaluation research · Benchmarking
Service composition

1 Introduction

Workflow management systems (WfMSs) are a core middleware technology for engineering service-oriented applications and for building service orchestrations [21]. Recently, WfMSs are being adapted to cloud computing environments to enable the development of scalable and elastic service-oriented systems.

The most critical part of a WfMS is probably the language in which a user can implement applications and services (i.e., workflows), that run on top of a WfMS. Selecting an unsuitable language or one that cannot easily be transferred to another system can have severe implications for a user, such as an inability to express business requirements or vendor lock-in [2]. This situation has been addressed by global standardization consortia. To this end, several organizations proposed workflow standards, as for example OASIS with the Web Services Business Process Execution Language (WS-BPEL) [20], or OMG with the Business Process Model and Notation (BPMN) [16].

Workflow standards [16, 20] define the language that can be used to implement workflows and the lifecycle of workflow instances. They are meant to clarify the exact scope, building blocks, constraints, and semantics of the language in a precise and unambiguous fashion. As a result, users may select a WfMS with respect to the standard it supports. Unfortunately, in many real-world systems, this assumption is flawed. For example: (1) the quality of standards is often not as high as expected. Especially the BPMN 2.0 standard has been shown to contain many inconsistencies, ambiguities, and editorial flaws [2, 9]. (2) A certification process is not available for any of the standards. As a consequence, any WfMS vendor can claim compliance to a standard without providing proof of this claim. Thus, many vendors just claim support for a standard [8]. (3) Even in the cases where standards provide an unambiguous specification, WfMSs do not necessarily follow that specification, but only implement parts of it [8, 12]. The standard-support related flaws, also lead to performance evaluation pitfalls in WfMSs, such as: (1) the execution performance of the same workflows differs significantly between WfMSs [22] and (2) the miscellaneous usage and implementations of workflow languages are obstacles for the construction of a standard benchmark [23].

As a consequence of the aforementioned flaws, the selection of a suitable WfMS constitutes a challenging task in practice. This paper aims to provide guidance for practitioners (e.g., users and vendors of WfMSs) by highlighting key issues during WfMS evaluation. The material is a cumulative report based on findings we collected over a period of more than five years of experience and derived through independent research initiatives regarding WfMSs¹. The novelty in this work comes from the aggregation of these results in a joint fashion as lessons learned and we aim at answering the following research questions:

RQ1 Which are the most common expectations and pitfalls during the usage of standard-based WfMSs for automated service composition?

RQ2 How does a practitioner experience the consequences of these pitfalls, in particular regarding the behavior and performance of the WfMS, and how can the pitfalls be addressed?

To answer these questions and to support an easy understanding and transfer to practice, we formulate the aggregated knowledge as lessons learned.

¹ BenchFlow - <http://benchflow.inf.usi.ch> and Betsy - <https://github.com/uniba-dsg/betsy>.

We mainly look at situations in which a WfMS is integrated into a more complex environment and is used as a service by many other systems.

This paper is based on an extended abstract [5], which motivates the reporting of lessons learned and briefly mentions a set of five lessons. Here, we report on an extended set of lessons learned, providing additional evidence and details gained with a larger set of WfMSs.

In the next section, Sect. 2, we discuss related work. Thereafter, in Sect. 3, we outline the process followed to derive lessons learned, the schema for their presentation in this paper, and the set of resulting lessons. Section 4 concludes the paper with a summary and an outline for future research directions.

2 Related Work

In this section, we discuss work related to our lessons learned, related workflow languages, and benchmarking approaches.

Two of the most prominent languages for modeling and executing workflows related to services and also relevant to our work here are WS-BPEL [20] and BPMN 2.0 [16]. The two languages serve similar goals but differ in their expressive power and the language constructs they provide: WS-BPEL is dedicated to the orchestration of web services [20], whereas BPMN 2.0 supports service invocations and message exchanges, but also human activities [16]. Here, we focus on automated workflows only and the human aspects of BPMN 2.0, such as collaborative process modeling, are deliberately out of scope.

Closely related to this paper is the work by Bianculli et al. [1]. The authors propose SOABench as an approach for evaluating the performance of WS-BPEL WfMSs. Here, our focus is broader, since we also take BPMN 2.0 into account and are not limited to the evaluation of performance efficiency aspects only, but cover other characteristics of software quality as well. Furthermore, Wohed et al. [27] evaluate several WfMSs and older versions of BPMN and BPEL for their support for workflow patterns. We also leverage workflow patterns in our work [8, 12, 22], but evaluate newer versions of said workflow languages. Similarly, Garcês et al. [7] present a survey of open source WfMSs. However, the authors address the problem from a different direction. They derive comparison criteria *a priori* from the workflow reference model and evaluate these criteria. The key difference to this work is that we present lessons learned, i.e., we did not derive *a priori* criteria but did a post-hoc study based on the observations made during our evaluations. Moreover, the set of systems evaluated is quite different from the systems we consider here. The same applies to Delgado et al. [3] who also build on similar quality models as we do here, but take a more generic stance. The authors state themselves in their study that they differ from our work in many aspects, such as the systems to be evaluated. Being similar to benchmarking methods and tools, approaches for test generation and testbed generation for service-oriented systems, such as [19], are related to our work. For instance, López et al. [19] propose a framework for black-box and property-based testing of web services. Although this framework is intended to test actual applications running on WfMSs, it could be leveraged to evaluate WfMSs as well.

3 Findings and Lessons Learned

A commonly accepted classification of research papers in software engineering is presented in the work by Wieringa et al. [26]. Using this classification, our work qualifies as *evaluation research*. More precisely, we aim at increasing knowledge by combining existing findings derived from multiple research groups into a common view. This can be achieved by a joint specification of *lessons learned* [26, p. 105].

The *lessons learned* presented in this work are based on cumulative research results, data collection, and experience derived by using and benchmarking different workflow language standards and WfMSs in a number of empirical studies [4, 6, 8, 9, 11–14, 17, 22, 23]. We conducted these studies over a period of several years independently of each other in separate groups, different environments, and with a different evaluation focus. To generate the lessons learned presented here, the group of authors met in person and in video conferencing sessions over an extended period of time. We discussed key lessons that we learned independently through our research in WfMSs evaluation and captured the ones that we learned jointly. The resulting list of lessons was prioritized through a joint voting process and the top lessons are presented in this paper. This research process is very similar to the way in which pattern languages are being developed.

The BPMN 2.0 WfMSs we consider in this paper are *Activiti*, *Bonita*, *Camunda*, and *jBPM*, which according to the vendor’s websites, are widely used in the industry. In total, we attempted the evaluation of 47 BPMN 2.0 WfMSs, but most of them could not be integrated into our evaluation approaches due to various reasons such as licensing issues, missing standard compliance, or the unavailability of management APIs [8]. For WS-BPEL, our lessons learned are based on the usage of *Apache ODE*, *OpenESB*, *bpel-g*, *Orchestra*, *Petals ESB*, and three commercial WfMSs whose names we are not allowed to disclose due to licensing. We decided to include pseudonymized results for the commercial WfMSs in this paper, although we acknowledge that this is less informative. The data collected are publicly available on an interactive dashboard². To continuously survey the standard-compliant WfMSs landscape, we created a Wikipedia page for BPMN 2.0 WfMSs³ which is kept up-to-date by public contributors and ourselves. In addition to helping practitioners in their decision-making, the dashboard and the Wikipedia pages help with improving transparency by reporting the actual state of the WfMSs ecosystem back to the vendors. To document the lessons learned, we use the following structured schema:

Expectation: Users of WfMSs have a number of expectations that are often perceived as true in practice and that are communicated in this fashion by WfMS vendors.

Observation: Although many of the expectations towards WfMSs can be fulfilled, some cannot. This also applies to aspects of considerable importance.

² <http://peace-project.github.io>, last visited at September 27, 2017.

³ <https://en.wikipedia.org/?curid=43305615>, last visited at September 27, 2017.

In the observations sections, we present and discuss existing evidence that an expectation is not met. Furthermore, we report cases in which it is met. *Consequence:* Not meeting a certain expectation has consequences, which are discussed in this part of the schema.

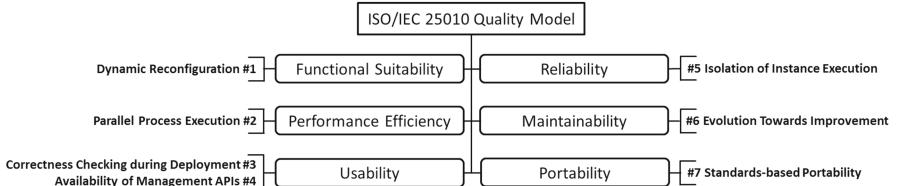


Fig. 1. Findings categorized using the ISO/IEC 25010 Quality Model

The final list of lessons identified in this study comprises seven items. For some of them, we identified WfMSs that avoid the pitfalls by following what we consider a good approach. For structuring purposes, we categorized these lessons using the ISO/IEC 25010 quality model [15] as shown in Fig. 1. According to this quality model, the lessons we defined can be grouped with respect to *functional suitability*, *performance efficiency*, *usability*, *reliability*, *Maintainability*, and *portability*. The ISO/IEC 25010 standard provides two more categories, namely, *compatibility* and *security*, that were not considered in this work, since we did not experience pitfalls specific to these categories. More specifically, a dedicated evaluation of security properties is not part of our initiatives and not planned for future work. Nevertheless, an evaluation dedicated to this property might very well discover new pitfalls.

3.1 Functional Suitability Findings

Lesson 1: Dynamic Reconfiguration

Expectation: WfMSs are expected to support dynamically changing environments and flexible dynamic service bindings [25]. This entails the reconfiguration of workflow instances at runtime when they interact with late-bound external systems and alternative service providers. It should be possible to pass the address of a service to a workflow instance and the instance should be able to redirect its communication channels to this service. As an example, in a workflow that processes orders, a buyer could send the address of his own web service to the workflow instance so that he will be notified when the order is complete.

Observation: In WS-BPEL, dynamic reconfiguration is possible in a standard conformant way, by updating the endpoint of specified partner links. However, one out of three proprietary WfMSs evaluated in [12] and four out of five open source WfMSs, namely, Apache ODE⁴, OpenESB, Orchestra, and Petals ESB,

⁴ The developers of ODE fixed this in a later version.

evaluated in [11, 12] do not support standard-conformant dynamic reconfiguration and only bpel-g does so. We did not check non-standard vendor-specific extensions which could provide similar functionality. In BPMN 2.0, dynamic reconfiguration is defined only in an abstract way, but no WfMS actually supports it based on the standard.

Consequence: Because of the lack of dynamic reconfiguration, the creation of self-adapting systems is hindered in WS-BPEL WfMSs. Instead of changing the channels within the workflow, each external web service needs to be encapsulated through a proxy service which itself can be dynamically reconfigured, leading to higher development and maintenance costs [25]. In BPMN 2.0 WfMSs, it is not possible to implement dynamic reconfiguration in a standards-based fashion.

3.2 Performance Efficiency Findings

Lesson 2: Parallel Process Execution

Also affects the functional suitability.

Expectation: One of the major advantages of workflow languages is that one can specify the control-flow of workflows in a declarative way. Then, any WfMS that implements the execution semantics of the workflow language should execute instances of these workflows by following the specified control-flow definitions. When modeling parallelism, the user expects that the underlying WfMS will execute the constructs that are marked as parallel to each other in a concurrent manner. Parallelism can be expressed with various control flow constructs in both standards. Using WS-BPEL [20] it is possible to define parallel execution using the *forEach* and *flow* constructs. Moreover, *eventHandlers* are always executed in parallel for a workflow scope that is attached to them. BPMN 2.0 [16] provides a similar variety of constructs for expressing parallelism. The most commonly used one is the splitting *parallelGateway* that acts as a fork operation and causes the parallel execution of the connected branches. Other possibilities are the use of *inclusiveGateways*, *eventBasedGateways*, or the definition of *MultiInstanceLoopCharacteristics* for tasks and sub-processes.

Observation: In [11, 12], we studied the behavior of WS-BPEL [20] workflow models containing the parallel *forEach* element. The research was conducted for five open source (Apache ODE, bpel-g, OpenESB, Orchestra, and Petals ESB) and three proprietary WS-BPEL WfMSs. The results showed that two open source (OpenESB, Petals ESB) and one proprietary WfMSs ignored the parallel semantics on the *flow* and *forEach* elements and one (Orchestra) on the *forEach* element only, *silently* resulting in a sequential execution.

Likewise, we evaluated three open source BPMN 2.0 WfMSs [22], namely Camunda, Activiti and jBPM regarding their support for workflow patterns. These patterns are considered as pieces of functionality that should be easily expressible in any workflow language. We tested the WfMSs against five fundamental control-flow workflow patterns, two of which contain parallelism. The results show that all benchmarked WfMSs implement parallelism in a pseudo-parallel, non-deterministic way, which is also reported in another study [8].

jBPM uses a random execution order of the parallel elements, while Camunda and Activiti always execute the parallel elements in the order in which they are defined in the workflow model. Moreover, jBPM shows a significant drop in performance if the parallelism construct is used [22].

Consequence: Due to pseudo-parallel execution, it is not possible to speed up the execution of independent control-flow branches by using the language constructs dedicated to parallelism. Apart from the performance aspect, this is also problematic as in some situations the functional correctness might depend on a truly parallel execution. For instance, several workflow patterns, such as multiple instances without a priori runtime knowledge, build on truly parallel execution, and thus, “the lack of truly parallel execution in a WfMS is the biggest obstacle to pattern support” [12, p. 111].

3.3 Usability Findings

Lesson 3: Correctness Checking during Deployment

Expectation: Both workflow languages, WS-BPEL [20] and BPMN 2.0 [16], define constraints regarding the correctness of modeled workflows. WS-BPEL explicitly lists 94 rules named *static analysis rules*, which describe issues that should be detected by any standard compliant WfMS. Thus it is to be expected that WfMSs are capable of detecting invalid workflow models at deploy time.

Observation: Deploying invalid workflows that violate static analysis rules or BPMN 2.0 constraints revealed that most WfMSs are not capable of this kind of detection [8, 14]. Regarding WS-BPEL WfMSs, we evaluated several systems for their coverage of static analysis rules [14]. Ignoring the reference implementation of BPEL which we do not consider here, we found that a single WfMS (OpenESB) performs no detection at all, and the rest have a highly varying detection rate of at most 75%. For BPMN 2.0, a common omission can be found in the missing validation of *timer* conditions. None of the three WfMSs benchmarked in [22], namely Camunda, Activiti, and jBPM, validated *timer* conditions at deploy-time, although there is a mandatory definition of the format [16].

Consequence: As invalid workflows are not rejected on deployment, errors are not detected early in the development process. Thus, errors may be hidden for a long time in production use only to be found later, which is costly. Invalid workflows may fail at runtime creating runtime errors in the WfMSs. To make things worse, in some cases the workflows do not crash observably, but instead complete with non-deterministic results. If the used WfMS is weak in detecting violations of standard-defined rules, the users creating workflows for deployment should consider using external tools for validating the workflows prior to deployment.

Lesson 4: Availability of Management APIs

Expectation: In production, WfMSs are usually part of a more complex software ecosystem and typically interact with other services. They are also integrated more and more into a continuous integration and delivery lifecycle [24].

It is to be expected that WfMS vendors provide management APIs [4] to support continuous integration and delivery.

Observation: Only four (Camunda, Bonita, jBPM, and Activiti) out of a set of 47 BPMN 2.0 WfMSs which we analyzed allow an automatic deployment and execution of workflows through a REST API [8]. Camunda and Activiti expose complete REST APIs to the clients so that the interaction with those WfMSs is straightforward. Bonita and jBPM provide partial support for interaction through the provided REST APIs. For example, the former misses the possibility to log into the API, forcing the user to log in using the so-called Web REST API⁵, while the latter misses an API to deploy workflows, leading to the need for a workaround. Out of the remaining 43 WfMSs, many require human interaction with a user interface at various stages. Examples range from a manual import of standard-compliant BPMN 2.0 workflows prior to deployment, over manual deployment using a web front-end, to manual creation of new workflow instances. In contrast, the WfMSs supporting WS-BPEL do not support REST APIs, but require file handling or other APIs (ODE, bpel-g, Orchestra, Petals ESB) [10, 11]. One (OpenESB) even lacks a remotely accessible API.

Consequence: Given the limitations or lack of the WfMSs' APIs, it is often hard or impossible to integrate the products in a fully automated continuous integration lifecycle. For instance, it is often not possible to quickly detect errors introduced in revisions of existing workflows by automated tests. This hinders the application of agile development methods that rely on short feedback loops [24].

3.4 Reliability Findings

Lesson 5: Isolation of Instance Execution

Expectation: Workflow instances should be executed in a sandbox. Users expect that instances cannot influence each other only because they are executed in the same environment. Moreover, faulty instances should not have an impact on the stability and integrity of the WfMS itself. Facilities need to be in place to restrict hostile instances from breaking out of their runtime environment, overloading the WfMS performance-wise, or taking down the entire WfMS [13, 18]. Furthermore, the WfMS should also try to minimize performance interference of, and between, different workflow instances [18].

Observation: For some Apache ODE versions, we observed that individual workflow instances are able to jeopardize the execution of other workflow instances or even severely affect the underlying WfMS stability. If a process instance enters a state of busy waiting, the resulting CPU load crashes the whole WfMS [12]. In the case of BPMN 2.0 WfMSs, we found that single versions of Activiti, jBPM, and Camunda were crashing due to memory leaks if the executed workflows are using infinite loops to execute script tasks. Moreover,

⁵ <http://documentation.bonitasoft.com/?page=rest-api-overview#toc2>, last visited at September 27, 2017.

the default configurations of Activiti and Camunda were not stable when using workflows containing loops and 1500 concurrently interacting users [22].

Consequence: The effect of missing isolation during workflow instance execution is similar to a single process crashing a complete operating system. It is obvious that this should not happen. The WfMSs should be safe from possible crashes of workflows instances, protecting other running instances. This can be achieved, for example, by detecting excessive resource usage (e.g., RAM, CPU, I/O) and suspending the critical workflow instances. Another complementary approach could be the usage of independent WfMS installations for different, independent sets of workflows instances. This is facilitated as the WfMSs vendors are starting to support virtualization techniques such as Docker containers.

3.5 Maintainability Findings

Lesson 6: Evolution Towards Improvement

Also affects performance efficiency and functional suitability.

Expectation: WfMSs should improve and evolve over time towards a higher degree of maturity. When upgrading to a newer version, users expect that it will be better than the previous one. For WfMSs, usual expectations are improvements in (i) functionality available to the users, (ii) number of language features supported, (iii) performance and (iv) reductions of workflow instances execution cost.

Observation: In [8], we investigated the evolution of the three BPMN 2.0 WfMSs Camunda, Activiti, and jBPM over the period of three years. All the three WfMSs evolved in terms of functionality available. Besides the evolution in functionality, we also discovered regressions over the years. For example, in the cases of Activiti and jBPM there were features that stopped working after upgrading to the next release. What is more, all three WfMSs made only marginal progress in the support of more BPMN 2.0 features. Also, research in performance benchmarking of the Activiti and Camunda WfMSs revealed that over the versions the performance of the WfMSs decreased [6]. In particular, the time needed by the WfMSs to execute single workflow instances constantly increased over time, by approximately 8% per year.

Consequence: There are two consequences for users: (1) they should be careful when upgrading to a newer version as regressions (in terms of supported features and performance) are possible, and (2) users should not expect that language feature support (especially for BPMN 2.0) will increase from version to version. This creates the necessity for users to benchmark WfMSs extensively before deciding about adopting newer versions.

3.6 Portability Findings

Lesson 7: Standards-based Portability

Expectation: One of the goals of standardizing workflow languages and WfMSs is to establish a commonly agreed set of functionality and a serialization format for specifying the workflows that enables their portability [16, 20]. Given that a standard is supported by multiple WfMSs, users should be able to move workflows implemented in the standard between any of these systems. This protects from vendor lock-in. Moreover, the execution semantics of a standards-based workflow should be identical on any WfMS supporting the standard.

Observation: In the absence of certification authorities, standard acronyms are rather hollow. For instance, there are many vendors that state to support BPMN 2.0. In [8], we tested 47 products stating to implement BPMN 2.0 and discovered that only three of them (Activiti, Camunda, and jBPM) were able to import, deploy, and execute workflows expressed directly in the standardized execution format. Instead, many vendors rely on custom serialization formats, as for example Bonita⁶, and only provide a subset of the visual BPMN 2.0 shapes for modeling.

Even if execution using a standardized language is supported in principle, this support is often limited to selected features of the language. Details of feature support are reported in [11, 12] for WfMSs using WS-BPEL and in [8] for WfMSs using BPMN 2.0. Essentially, for both standards, the number of language features that is commonly supported by a large majority of the tested WfMSs is limited to around 40% of the features defined in the respective standards. The supported features are limited to the basic part of the standards, such as enabling sequential execution, conditional branching, and basic looping.

Consequence: Modeling a workflow in compliance to a standard does not guarantee that the workflow can be executed by a WfMS. The situation is further complicated by the fact that some vendors serialize workflows with custom language extensions specific to their product, which introduces vendor lock-in.

3.7 Summary

We presented seven lessons for which we formulated common expectations and discussed good approaches and potential pitfalls [RQ1]. In Table 1, we summarize these lessons learned. The table reports the findings for all the WfMSs providing sufficient APIs [4], as discussed in lesson 3.3, enabling us to automate the analysis producing the data on which most of the findings are based on. We use a trivalent rating that classifies WfMSs as using a good approach (+), containing a pitfall (-), or partially containing a pitfall (~). As evident from the table, each WfMS has advantages and disadvantages and no single WfMS provides a good approach for all of the discussed lessons.

⁶ <http://documentation.bonitasoft.com/?page=build-a-process-for-deployment>, last visited at September 27, 2017.

Table 1. Summary of the Findings: + (good approach), – (pitfall present), ~ (pitfall partially present), N/A (no observations)

WfMS	1. Dynamic Reconfiguration	2. Parallel Execution	3. Correctness Checking	4. Management APIs	5. Instance Isolation	6. Improved Evolution	7. Standard Portability
BPMN							
<i>Activiti</i>	N/A	– [8, 22]	~ [8, 22]	+ [8]	~ [22]	~ [6, 8]	~ [8]
<i>Bonita</i>	N/A	N/A	N/A	~	N/A	N/A	–
<i>Camunda</i>	N/A	– [8, 22]	~ [8, 22]	+ [8]	~ [22]	~ [6, 8]	~ [8]
<i>jBPM</i>	N/A	– [8, 22]	~ [8, 22]	~ [8]	+ [22]	~ [8]	~ [8]
WS-BPEL							
<i>ODE</i>	+ [11]	+ [12]	~ [14]	+ [10]	– [12]	N/A	~ [11]
<i>OpenESB</i>	– [11]	– [12]	– [14]	– [10]	+ [12]	N/A	~ [11]
<i>bpel-g</i>	+ [11]	+ [12]	~ [14]	+ [10]	+ [12]	N/A	~ [11]
<i>Orchestra</i>	– [11]	~ [12]	~ [14]	+ [10]	+ [12]	N/A	~ [11]
<i>Petals ESB</i>	– [11]	– [12]	~ [14]	+ [10]	+ [12]	N/A	~ [11]
<i>3 Commerc.</i>	~ [12]	~ [12]	N/A	N/A	+ [12]	N/A	~ [12]

The aggregated results highlight the relationship between workflow standards, WfMSs, and the user expectations that follow from this relationship. To answer RQ1, we identified pitfalls in the areas of functional suitability, performance efficiency, usability, reliability, maintainability, and portability. Despite the existing standards, expectations about the available functionality or portability of workflow models are often not fulfilled. The usability of WfMSs in terms of correctness checking and available APIs is limited in many cases and new versions do not necessarily lead to improvements. Lastly, scalability and workflow instance isolation can be problematic.

As consequences for users, referring to RQ2, this study shows that thorough research and evaluation before selecting a WfMS is inevitable despite the existence of workflow standards. It is unlikely to find a system that actually supports a complete standard and the danger of vendor lock-in is still real. Even when only updating a WfMS, a careful evaluation is needed because of potential regressions in terms of supported features or performance characteristics.

4 Conclusion and Future Work

In this paper, we presented a catalog of lessons learned regarding the usage of WfMSs, synthesized from findings obtained by two independent research initiatives over a five-year period. This synthesis uncovers a number of common expectations and pitfalls in WfMSs usage. No WfMS we investigated follows only good approaches, but they all lack important aspects. Most prominently, the standard conformance, and thus the portability of workflows, is severely hampered, despite the claimed support for standards. In this regard, it can be diagnosed that standards have largely failed their target of a harmonized WfMSs landscape. Ultimately, a WfMS selection will require a prioritization approach that ranks features according to their importance for the intended specific usage scenario. Based on such a ranking, the presented results can support the selection.

In conducting this study, we also learned lessons about aggregating benchmarking results as lessons learned. We have found the aggregation of lessons feasible and valuable despite the fact that our initiatives were focused on different quality attributes, i.e., conformance and performance, even if this meant we had to discard lessons that were important in one area, but not present in the other.

References

1. Bianculli, D., Binder, W., Drago, M.L.: Automated performance assessment for service-oriented middleware: a case study on BPEL engines. In: 19th WWW, Raleigh, North Carolina, USA, pp. 141–150, April 2010
2. Börger, E.: Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. *Softw. Syst. Model.* **11**(3), 305–318 (2012)
3. Delgado, A., Calegari, D., Milanese, P., Falcon, R., García, E.: A systematic approach for evaluating BPM systems: case studies on open source and proprietary tools. In: Damiani, E., Frati, F., Riehle, D., Wasserman, A.I. (eds.) OSS 2015. IAICT, vol. 451, pp. 81–90. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17837-0_8
4. Ferme, V., Ivanchikj, A., Pautasso, C., Skouradaki, M., Leymann, F.: A container-centric methodology for benchmarking workflow management systems. In: 6th CLOSER, Rome, Italy (2016)
5. Ferme, V., Lenhard, J., Harrer, S., Geiger, M., Pautasso, C.: Workflow management systems benchmarking: unfulfilled expectations and lessons learned (extended abstract). In: 39th ICSE Companion, Poster Track (2017)
6. Ferme, V., Skouradaki, M., Ivanchikj, A., Pautasso, C., Leymann, F.: Performance comparison between BPMN 2.0 workflow management systems versions. In: Reinhartz-Berger, I., Gulden, J., Nurcan, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2017. LNBP, vol. 287, pp. 103–118. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59466-8_7
7. Garcés, R., Jesus, T., Cardoso, J., Valente, P.: Open source workflow management systems: a concise survey. In: BPM & Workflow Handbook. Future Strategies (2009)
8. Geiger, M., Harrer, S., Lenhard, J., Wirtz, G.: BPMN 2.0: the state of support and implementation. *Future Gener. Comput. Syst.* **80**, 250–262 (2017)
9. Geiger, M., Wirtz, G.: BPMN 2.0 serialization - standard compliance issues and evaluation of modeling tools. In: 5th EMISA, September 2013
10. Harrer, S., Lenhard, J.: Betsy-a BPEL engine test system. Technical report 90, Otto-Friedrich Universität Bamberg, July 2012
11. Harrer, S., Lenhard, J., Wirtz, G.: BPEL conformance in open source engines. In: 5th IEEE SOCA, pp. 237–244, December 2012
12. Harrer, S., Lenhard, J., Wirtz, G.: Open source versus proprietary software in service-orientation: the case of BPEL engines. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 99–113. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_8
13. Harrer, S., Nizamic, F., Wirtz, G., Lazovik, A.: Towards a robustness evaluation framework for BPEL engines. In: 7th IEEE SOCA, pp. 199–206, November 2014
14. Harrer, S., Preißinger, C., Wirtz, G.: BPEL conformance in open source engines: the case of static analysis. In: 7th IEEE SOCA, pp. 33–40, November 2014

15. ISO/IEC: ISO/IEC 25010:2011; Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models (2011)
16. ISO/IEC: ISO/IEC 19510:2013 - Information technology - Object Management Group Business Process Model and Notation (2013). v2.0.2
17. Lenhard, J., Wirtz, G.: Portability of executable service-oriented processes: metrics and validation. *Serv. Oriented Comput. Appl.* **10**(4), 391–411 (2016)
18. Leymann, F.: BPEL vs. BPMN 2.0: should you care? In: Mendling, J., Weidlich, M., Weske, M. (eds.) BPMN 2010. LNBP, vol. 67, pp. 8–13. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16298-5_2
19. López, M., Ferreiro, H., Francisco, M.A., Castro, L.M.: Automatic generation of test models for web services using WSDL and OCL. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 483–490. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_37
20. OASIS: Web Services Business Process Execution Language (2007). v2.0
21. Peltz, C.: Web services orchestration and choreography. *Computer* **36**(10), 46–52 (2003)
22. Skouradaki, M., Ferme, V., Pautasso, C., Leymann, F., van Hoorn, A.: Micro-benchmarking BPMN 2.0 workflow management systems with workflow patterns. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 67–82. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_5
23. Skouradaki, M., Roller, D.H., Leymann, F., Ferme, V., Pautasso, C.: On the road to benchmarking BPMN 2.0 workflow engines. In: 6th ACM/SPEC ICPE, pp. 301–304. ACM (2015)
24. Thiemich, C., Puhlmann, F.: An agile BPM project methodology. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 291–306. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_25
25. Tsai, W.T., Song, W., Paul, R., Cao, Z., Huang, H.: Services-oriented dynamic reconfiguration framework for dependable distributed computing. In: COMPSAC, pp. 554–559 (2004)
26. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *RE* **11**(1), 102–107 (2006)
27. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006). https://doi.org/10.1007/11841760_12



Sustainable WAsTe Collection (SWAT): One Step Towards Smart and Spotless Cities

Daniel J. Rosa-Gallardo¹, Guadalupe Ortiz¹⁽⁾, Juan Boubeta-Puig¹,
and Alfonso García-de-Prado²

¹ Department of Computer Science and Engineering, University of Cádiz,
Avda. de la Universidad de Cádiz 10, 11519 Puerto Real, Cádiz, Spain
dani.rosagallardo@alum.uca.es,

{guadalupe.ortiz, juan.boubeta}@uca.es

² Department of Computer Technology and Architecture, University of Cádiz,
Avda. de la Universidad de Cádiz 10, 11519 Puerto Real, Cádiz, Spain
alfonso.garciaadeprado@uca.es

Abstract. For decades, urban areas have been faced with the huge challenge of waste disposal and collection. Even though developed countries have tackled this matter from multiple perspectives, some issues remain unsolved when talking about sustainable smart cities. In particular, waste collection and transportation routes are mostly planned statically, without bearing in mind real-time container capacity, as well as other unexpected events which might be of interest – for example, a blocked container lid or fire inside the container. In this paper, we propose a service-oriented sustainable solution to these problems based on the use of novel technologies which will provide us with real-time information and instant notifications of any issues related to waste containers and their collection. This way, thanks to the offered real-time services for the Internet of Things, local governments and official bodies will save resources and time. Even more, citizens will also have easy access to such services and information and therefore will benefit from real-time awareness about the container status.

Keywords: Sustainable waste management · Prevention · Internet of Things
Sensors · Service-oriented architecture · Mobile application

1 Introduction

Currently, there is a great focus on the development of smart cities and, specifically, sustainable smart cities. Not only is it a current day research area, but it is also a priority for governments, environmental protection agencies and even European commissions [1]. Furthermore, citizens have become aware of the importance of living in a sustainable world, but they need the means to be able to cooperate in the process.

Within this field, waste collection topic has always been socially excluded. Due to the unpleasant sight waste containers might present, many towns are replacing them by underground ecological islands with waste collection facilities. These solve the problem of bad odors that were present near a traditional waste container. However,

underground containers do not solve the problem when waste is not collected frequently enough, or when there is an unexpected amount of waste disposal.

In general, whichever type of container is used, there are frequent unpleasant and even dangerous situations due to waste disposed outside the container. The reason is that the mechanism established by councils for waste collection mostly consists of a fixed weekly schedule with a fixed route to be followed every day, regardless of how full or empty the containers are; besides, there is almost no monitoring on waste containers. This causes three main problems, among others: (1) money is wasted when collecting waste from empty or almost empty containers; (2) both residents and neighborhoods are affected when not collecting waste from full containers; (3) dangerous or unsuitable situations are not detected.

In the past some works about forecasting models for waste collection were published [2]; however forecasting is not always accurate and do not provide information about unpredictable facts. Now, some emerging proposals can be found on waste containers' fill level monitoring [3, 4]: they mostly focus only on providing optimized routes to the collection company office. Nevertheless, they do not provide additional real-time alerts to other interested parties nor dynamic updated routes to the track driver application based on alerts which might happen when the track is already on the way. To solve this problem, we propose SWAT: an Internet-of-Things (IoT)-based hardware infrastructure and software architecture for Sustainable WAsTe Collection. SWAT will monitor and provide us with waste islands information in real time and will send notifications about their status to any interested parties, such as the Fire Brigade. Please note that even though we refer to containers in ecological islands throughout the paper, the proposal is applicable to any type of waste container.

The remainder of the paper is organized as follows: First of all Sect. 2 explains the required background on Event-Driven Service-Oriented Architecture (ED-SOA) and Complex Event Processing (CEP) to facilitate the understanding of this paper, and presents related work. Then our solution SWAT is described in detail in Sect. 3. Afterwards SWAT evaluation and discussion are presented in Sect. 4 and 5, respectively. Finally, conclusions are drawn in Sect. 6.

2 Background

In this section, the software technologies particularly relevant for this paper scope are explained, as well as an overview on other waste management solutions is given.

2.1 Event-Driven Service-Oriented-Architecture

Service-oriented architecture (SOA) is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces [5]. The essential goal of a SOA is to enable general-purpose interoperability among existing technologies and extensibility to future enhancements. In particular, in this paper we will make use of REpresentational State Transfer (REST) services [6].

On the other hand, Event-Driven Architectures (EDAs) promote the detection of events and the subsequent intelligent reaction to them [7]. Currently, the integration of EDA and SOA is known as Event-Driven SOA or SOA 2.0. SOA 2.0 will ensure that services do not only exchange messages between them, but also publish events and receive event notifications from others. For this purpose, a highly distributable communication and integration backbone is required. This functionality is provided by an Enterprise Service Bus (ESB), a middleware application that provides interoperability between different communication protocols and which can be used as an integration platform that enables existing applications to be exposed as services [5]. Mule [8] is one well appreciated ESBs because of its integration with cloud platforms and multiple tools and scenarios. SOA 2.0 leans on CEP, a technology that allows the analysis and correlation of large volumes of data as explained subsequently.

2.2 Complex Event Processing

CEP is a cutting-edge technology which provides powerful techniques for processing and correlating events to detect relevant situations (complex events) in real time. An event can be defined as anything that happens or could happen [9]. We can have simple events —they are indivisible and happen at a point in time— and complex events, which contain more semantic meaning summarizing a set of other events. Besides, events can be derived from other events by applying or matching event patterns, that is, templates where the conditions describing the situations to be detected are specified. Such event patterns are defined using specific languages developed for this purpose known as *Event Processing Languages* (EPLs). In this scope, a CEP engine is the software used to match these patterns over continuous and heterogeneous event streams, and to raise alerts about the complex events created when detecting such event patterns. In this paper we make use of Esper [10], which is a well-known open source CEP engine, written in Java, what facilitates its use in multiple platforms.

2.3 Summary of Existing Approaches

Some cities are starting to propose smart approaches in order to provide sustainable waste collection systems as is being done in one district of Los Angeles [11]. BigBelly company propose the use of their intelligent containers which compress the waste and send a message to the collectors' telephone when they need emptying [12]; currently a pilot is being performed in New York. Dugdhe et al. depict a proposal for suggesting a schedule for waste collection trucks [4]. More mature is Enevo research project funded by the European Commission's Horizon 2020 program [3], a solution combining fill-level monitoring and dynamic route planning to eliminate inefficiencies and costs associated with inflexible static waste collection routes. We can also find some mobile applications which encourage citizens to recycle providing them information about the waste type and where to find the appropriate container [13].

However, as far as we are aware, none of them has supplied appropriate infrastructure to provide real-time information not only about container fill level but also other unexpected events, such as a blocked container lid, unsuitable material disposal or a fire, which might be of great interest for governments and official bodies as well as

for the citizens. Besides, SWAT consider that the truck driver will carry a mobile application which will be updated in real time, even during the waste collection route, so any unexpected issue can be dealt with immediately. Moreover, existent solutions provide and notify their status to management offices only, whereas our proposal considers that citizens play a key role in the sustainable waste collection process.

3 SWAT: Sustainable WAsTe Collection

In this section, we are going to show a high-level overview of SWAT infrastructure and proposed architecture to easy the comprehension of the remaining subsections. Then we get into details: firstly we describe the hardware infrastructure, secondly we explain the management module and then the software clients are described; finally we make clear how alerts are detected and how actions are risen automatically.

3.1 SWAT Infrastructure and Architecture

Figure 1 represents a schematic view of the developed prototype for SWAT infrastructure. On the left hand side, we can see the board and sensors to be installed in waste containers; that is, the hardware requirements. On the right hand side, we can see the mobile devices which will receive a notification when a certain action is required regarding waste containers; that is, the software clients. Finally, the central part of the diagram shows all Internet-based communications and the management software module. The operation proceeds as follows:

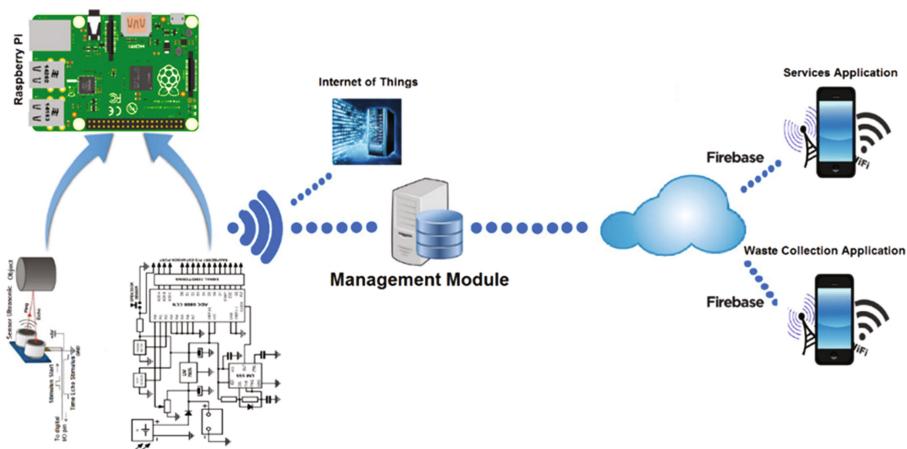


Fig. 1. Sustainable WAsTe Collection schematic view

1. Waste containers equipped with the hardware infrastructure and corresponding sensors will provide information concerning how full the container is, if any obtrusive element is blocking it, as well as information about the temperature and

air opacity inside the container. This information will be constantly sent through the Internet to the server where the management software module is installed. The information is also sent to an IoT platform where any interested party – a citizen or an environmental agency, for instance – can check container status in real time.

2. When the sensors' information from waste containers reaches the management software module, the latter processes and stores such information, detecting any situation which requires an action (such as a fire or a full container) in real time.
3. When actions are required, notifications and alerts are sent to the relevant management office/body (firefighters, waste collection company, et cetera).

Regardless of the special actions required on risen alerts, before the waste collection employee starts his daily collection, the system provides him with the optimal route according to container status information, which can be automatically updated during the route if any additional alert is detected.

3.2 Hardware Infrastructure

The aim of the hardware infrastructure implemented is to acquire and send the following data from each ecological island: temperature, unsuitable waste, fill level, blocked lid, smoke presence and battery status. For this purpose, we used a Raspberry Pi B+ as the central unit to be connected to the implemented circuit and plugged sensors (see left-hand side of Fig. 2). The Raspberry Pi obtains the sensor measurements and sends the data to the management software module every 10 min. The most relevant elements connected to the circuit are:



Fig. 2. Hardware infrastructure and container mock-up

- Temperature sensor: it measures the temperature inside the container. We have used TMP35 temperature integrated circuit, which provides an output voltage linearly-proportional to the Centigrade temperature of $10 \text{ mV}^{\circ}\text{C}$.
- Dust/Smoke sensor: it measures air opacity in the container. We have used sensor GP2Y1010AU0F which is an optical air quality sensor, designed to sense dust particles: it is based on an infrared emitting diode and a phototransistor which are diagonally arranged to allow the sensor to detect the reflected light of dust in air.
- Limit switch sensor: it detects container lid blockages. It is a limit switch connected to mass and to a pull up resistor.

- Ultrasonic transducer sensor: it detects how full the container is. We used Ultrasonic ranging module HC - SR04: sending a pulse input and measuring pulse signal back, we can calculate the waste level in the container.
- Lithium polymer (LIPO) battery: it provides energy to the circuit in case of power failure or cloudy weather. The 7.4v and 3200 mA LIPO battery used provides more than enough battery to feed the circuit, since the system load in text mode with the minimum required services and consumes less than 200 mA. Please note that the higher consume of the system is when sensors information is submitted; to minimize it we send the information from all sensors at the same point in time.
- Solar cell: it charges the battery. We used an 8.5 V solar cell, which feeds the system charging batteries when there is a feeding failure and at night. Also, we needed to use a LM7805 voltage regulator in between the solar cell and the LiPo battery.

Besides, as shown on the right hand side of Fig. 2, we implemented a scaled container mock-up and installed the software infrastructure on it in order to test the system's functionality.

3.3 Management Software Module

The architecture proposed and implemented for the administration software module is shown in Fig. 3. It is composed of an ESB, a REST Application Programming Interface (API), a CEP engine, an SQL Database, a Not-Only SQL (NoSQL) Database, an E-mail service and a Google Firebase service. Let us describe each element in detail:

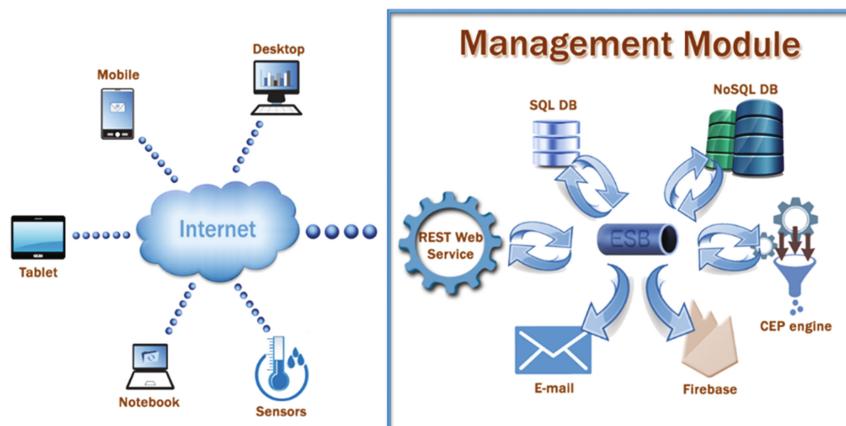


Fig. 3. Management module architecture

- ESB. As previously explained an ESB eases communication between several devices and software modules in heterogeneous systems [5]. In this approach, we use the ESB to transform and route the information reaching the system from the

sensors in the hardware infrastructure to the CEP engine and to the event consumers.

- REST API. This is a set of functions that can be invoked, according to REST principles. In this case, a REST API will be used as the interface to external software for the management module, i.e. other software can interact with the ESB through this API. Some of the services offered by the API are:
 - The collector truck employee will be able to obtain the optimal route for waste collection from his software client. Please bear in mind we use Google maps to provide the optimal route; our contribution is knowing -according to the real time container waste level- which containers should be visited.
 - The hardware infrastructure may post the data taken from all container sensors.
 - A citizen could get information about the closest container not damaged or full.
- CEP engine. As previously explained, the CEP engine allows us to analyze and correlate a high number of information events with the aim of detecting relevant or critical situations (alerts) in real time [9]. In this case, we detect when there might be a fire in the container and when it is full or blocked, based on the information provided by the sensors and routed to the CEP engine by the ESB, and we send the corresponding alert to the official body in charge.
- SQL Database. As could not be otherwise, we will use a relational database to store the information about local agencies, container location, alerts available in the system, et cetera.
- NoSQL Database. We will use it to store simple events coming from the hardware infrastructure (sensor measurements) and complex events coming from the CEP engine (detected situations of interest). The reason to have an additional NoSQL database is to facilitate the scalability of the system, so that a growing number of sensors connected to the system would not decrease its performance.
- An email service. Any type of alert will be sent to the licensed agency.
- Firebase Service. Firebase is a Google platform which provides several facilities for mobile applications (<https://firebase.google.com/>). In this case we make use of their facility for mobile notification submission. Mobile devices of the corresponding agency or official body will be sent a notification. Firebase could have also been used to store the relational database information, but we opted for an external database to keep it in the local server.

Therefore, the process is as follows:

1. Sensing data reach the system through the REST API.
2. These data flow through the ESB.
3. The sensing data are transformed into events: (a) events are sent to the CEP engine; (b) events are stored in the NoSQL database.
4. Alerts are detected in the CEP engine (according to previously predefined patterns):
 - (a) notifications/alerts are sent to the corresponding management office/body and corresponding actions are automatically taken; (b) complex events are stored in the NoSQL database.
5. The information in the databases can be consulted through the REST API.

3.4 Software Clients

Two Android mobile applications were developed: one for official bodies and another one for the maintenance and collector company, with the following uses in mind:

- Emergency and maintenance bodies (police, firefighters and maintenance staff) will have a mobile application which receives the Firebase alerts sent from the management software. In Fig. 4, the first and second image show the main menu from such application and a received alert, respectively.



Fig. 4. Official bodies and waste collector employee software client

- Maintenance alerts can be sent to the second mobile application, aimed at the waste collection licensed company. It is also used, for example, to obtain the optimal waste collection route, as shown in the fourth image of Fig. 4; the main menu of this application is shown in the third image.

3.5 SWAT Alerts and Actions

All ecological islands will send their data to the management module every 10 min. However, in order to avoid feasible fire or unsuitable container use among readings, a specific process will be checking their status every 5 s, in order to raise the alarm as soon as possible, if necessary. This means that if such data are sensed, the alert is sent to the management module at once.

Let us explain now what exactly our system will detect, how we define what we wish to detect and what will we do upon every triggered alert.

SWAT Alerts

The management module contemplates 4 critical situations, as summarized below:

- *PotentialDust* (Alert level 1): it detects unreasonable air opacity, that is, the density of dust or smoke in the air.
- *PotentialFire* (Alert level 2): it detects fire in the container.

- *PotentialLowBattery* (Alert level 3): it detects when the ecological island's battery level is not enough for proper functioning.
- *PotentialBlockedLid* (Alert level 4): it detects container lid blockages.

The definition of such situations to be detected in the CEP engine can be done manually by coding the patterns; alternatively, those who are not expert on programming languages but have knowledge about the matter of the patterns in question, can use our graphical intuitive interface created for this purpose, called MEdit4CEP [14]. This tool is capable of automatically transforming the modeled event patterns into Esper EPL code. Moreover, this interface allows us to deploy patterns into the system automatically: we only need to graphically define and add the new desired patterns. At <http://dx.doi.org/10.17632/z6pzv33xhy.1> we can find an example of how the event pattern for detecting potential low battery (*PotentialLowBattery*) in an ecological island has been modeled and automatically transformed into EPL code by using MEdit4CEP.

We would like to highlight that this management module can be extended to detect new critical situation types: we only need to implement the new patterns coding them manually or graphically with MEdit4CEP and deploying them in the CEP engine.

SWAT Actions

The actions to be taken when the explained alert situations are detected follow:

- Alert Level 1: inappropriate material disposal, such as construction waste. The police department is notified by a mobile notification using Firebase.
- Alert Level 2: there is fire in the container. The firefighters and police department are notified by a mobile notification using Firebase.
- Alert Level 3: low battery level. The maintenance service is notified by a mobile notification using Firebase.
- Alert Level 4: blocked container lid. The maintenance service is notified by a mobile notification using Firebase.

With the built mock-up, we have only been able to test a limited number of functionalities and situations. In order to test the system in depth and see if alerts were raised appropriately, we implemented a daemon that simulates the values taken from every sensor from several island and sends continuous data to the management module. Besides, we send it to an IoT platform with the purpose of making it available to additional clients and applications: this way, anyone could use these data in their software clients and citizens could check it from their Internet browsers, for instance to see whether the container where they are going to dispose of their garbage is full.

4 Evaluation

Our prototype is constantly sending data to a channel in ThingSpeak IoT platform; in particular this channel can be followed at <https://thingspeak.com/channels/72664>, where we can see our ecological island battery level, temperature, dust level, fill level and if the lid is blocked or not, as well as the location of the container.

In order to demonstrate the feasibility of the solution proposed in this paper, SWAT, we have carried out some performance tests concerning the amount of data supposed to

be managed by medium and big-sized cities around the world. The full software architecture was deployed in a workstation with i3-540 (4 M cache, 3.6 Ghz), 8 GB of RAM memory and SATA2 hard drive and we have used an emulator to generate the messages sent to the system. Since a medium-sized city can have around 5000 containers, we tested the system, receiving 5000 messages every 10 min – every message contains 5 sensor measurements. We used ESB and database consoles to monitor this simulated data in real time. We checked that the system perfectly supported this load of events and responded in less than a millisecond with the corresponding complex events detected as well as the corresponding notifications; thereby, the alerts were launched in due time according to the values entering the system. As a big-sized city can approximately have around 20000 or 30000 containers, we also tested the system managing 30000 messages every 10 min and this worked appropriately.

Regardless of the number of containers expected to be dealt with by the system, we proceeded with additional performance tests to check how the system managed bigger amounts of data. We tested the system sending big amounts of events and increasing the number of events per minute submitted to the system (test set 1, 2 and 3 with 13300, 26700 and 53800 events per minute, respectively). As we can see in Fig. 5, even though we increased the submission speed over 50000 events per minute as well as the total amount of submitted events; the system continued running perfectly, detecting the corresponding relevant events.



Fig. 5. Results of the performance tests

As a result, we consider the system is highly scalable. Of course, we could use enterprise editions rather than community ones for implementation software as well as more powerful workstations or even distribute several components of the system in different machines to scale it around the world.

5 Discussion

The proposed software architecture makes use of emerging technologies which provide the platform with significant advantages, namely (1) being able to process and analyze the data coming from several garbage ecological islands in real time, (2) being able to

therefore detect and notify the corresponding alerts to the interested parties in real time. There are further benefits from using this software platform:

- The system is highly scalable, what means that a garbage collection company having containers in multiple locations – in the same or different cities – might deal with all of them using the same platform.
- The fact of being able to be notified of the status of every container in real time implies a save of resources and budget for all the parties: the garbage collection company will optimize the routes and save on fuel and employees, consequently they will be able to offer better prices for their services to the local governments, which will therefore also save on their budget dedicated for these purposes. Moreover, the citizens will be happier with their local governments and with their cities. Besides, fires will be detected earlier, and notified to all the interested parties, avoiding further personal and material damages. Last but not least, avoiding the accumulation of garbage outside the containers, will avoid the subsequently insalubrity and unhealthiness for the citizens and city in general.
- The system is also easily extendable thanks to the use of the ESB and therefore it could be integrated in the future with other relevant rising approaches, for instance to improve the promotion of appropriate waste disposal [15], or to include additional prevention policies.
- Since the device to be installed in the container is considerably cheap (less than 10 euros plus the cost of the solar cell if power supply if not available), the company which is in charge of waste collection could save big money in fuel when rescheduling the route according to the necessities of the containers.

In our near future work, we will complete our software architecture with the integration of other event consumers, such as actuators, social network services and a mobile application for the citizens. These consumers will be easily integrated into our architecture thanks to the ESB, which helps to increase the flexibility of heterogeneous environments. On the one hand, the use of actuators will become a benefit since some mechanical actions could then be taken at ecological islands when detecting specific event patterns. For instance, the actuator motor would force the container mouth to close or open in two situations: (1) close the mouth when the container is full, to prevent citizens from trying to introduce more waste, which can lead to clogging of the hydraulic system openness, and (2) open the mouth to ventilate the container in case of fire. On the other, the integration of the architecture with social network services like Twitter will permit twitting the detected alerts in an account to which interested citizens might subscribe. Even more, the mobile application will permit the users to check and receive notifications about the status of the nearby containers at any time as well as to provide information about them to the system – a citizen might detect for instance some accumulation of waste put down around the container by an antisocial citizen, which would not be detected by the sensors.

6 Conclusion

SWAT is an Internet-of-Things (IoT)-based hardware infrastructure and software architecture for Sustainable WAsTe Collection that will save waste collection companies' money and will prevent neighborhoods from suffering incommodities caused by full or damaged waste containers. SWAT will monitor and provide us with waste container information in real time, as well as alerting maintenance and official bodies when actions are required. Therefore, SWAT can help to pave the way to future smart sustainable cities and can make citizens aware and involved in this long journey.

Acknowledgements. The authors thank the support from the Spanish Ministry of Science and Innovation and the European Union FEDER Funds through the project TIN2015-65845-C3-3-R and TIN2014-53986-REDT/TIN2016-81978-REDT.

References

1. European Commission: Market Place of the European Innovation Partnership on Smart Cities and Communities. <https://eu-smartcities.eu/>
2. Abbasi, M., El Hanandeh, A.: Forecasting municipal solid waste generation using artificial intelligence modelling approaches. *Waste Manag.* **56**, 13–22 (2016)
3. Enevo: Optimising Waste Collection. <https://www.enevo.com/>
4. Dugdhei, S., Shelar, P., Jire, S., Apte, A.: Efficient waste collection system. In: Presented at the 2016 International Conference on Internet of Things and Applications (IOTA), January 2016
5. Papazoglou, M.P.: Web Services and SOA: Principles and Technology. Pearson Education, Essex (2012)
6. Burke, B.: RESTful Java with JAX-RS 2.0. O'Reilly, Beijing (2013)
7. Taylor, H., Yochem, Y., Phillips, L., Martinez, F.: Event-Driven Architecture: How SOA Enables the Real-time Enterprise. Addison-Wesley, Boston, London (2008)
8. MuleSoft: Mule ESB. <http://www.mulesoft.org/>
9. Luckham, D.C.: Event Processing for Business: Organizing the Real-Time Enterprise. Wiley, Hoboken (2012)
10. EsperTech: Esper - Complex Event Processing. <http://www.espertech.com/esper/>
11. Los Angeles Department of Public Works: Waste Management. <http://dpw.lacounty.gov/landing/wasteManagement.cfm>
12. BigBelly: Big Belly Solar. <http://bigbelly.com/>
13. Bonino, D., Alizo, M.T.D., Pastrone, C., Spirito, M.: WasteApp: Smarter waste recycling for smart citizens. In: Presented at the International Multidisciplinary Conference on Computer and Energy Science (SpliTech), July 2016
14. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: MEdit4CEP: a model-driven solution for real-time decision making in SOA 2.0. *Knowl. Based Syst.* **89**, 97–112 (2015)
15. Guo, H., Hobbs, B.F., Lasater, M.E., Parker, C.L., Winch, P.J.: System dynamics-based evaluation of interventions to promote appropriate waste disposal behaviors in low-income urban areas: a Baltimore case study. *Waste Manag.* **56**, 547–560 (2016)



Designing Suitable Access Control for Web-Connected Smart Home Platforms

Sebastian Werner^(✉), Frank Pallas, and David Bermbach

Information Systems Engineering Research Group, Technische Universität Berlin,
Berlin, Germany
{sw,fp,db}@ise.tu-berlin.de

Abstract. Access control in web-connected smart home platforms exhibits unique characteristics and challenges. In this paper, we therefore discuss suitable access control mechanisms specifically tailored to such platforms. Based on a set of relevant scenarios, we identify requirements and available technologies for fulfilling them. We then present our experiences gained from implementing access control meeting the identified requirements in OpenHAB, a widely used smart home platform.

Keywords: Access control · IoT · Smart home

1 Introduction

The Internet of Things (IoT) has been finding more and more adoption in the last few years and is seeing continuous growth. Gartner, for instance, expects 20.4 billion connected “things” to be in use by 2020 [10]. A particularly popular area in the IoT are the so-called smart homes where sensors and smart appliances are connected using often rule-based approaches to increase comfort for the home inhabitants.

Today, smart home platforms like OpenHAB¹ often run in isolated networks not connected to the public Internet. However, we expect this to gradually get replaced by more open deployments that include external services on the web and in the cloud, e.g., if-this-then-that (IFTTT)² or Amazon’s Machine Learning service³, through web APIs, e.g., for weather data, or even using devices such as Google Home⁴ or Amazon Echo. For such deployments, however, access control is a crucial feature of the smart home platform both to protect the inhabitants’ privacy but also to protect them from malicious attacks [8, 14]. When the possibility of tampering with the physical world exists, security becomes even more important – some recent events underline this [2, 17].

¹ openhab.org.

² ifttt.com.

³ aws.amazon.com/machine-learning.

⁴ madeby.google.com/home.

While access control is obviously highly important in smart home platforms, there are a number of characteristics of such systems that make it hard or at least non-optimal to simply reuse access control strategies from other domains, e.g., from file systems or mail servers. Therefore, we make the following contributions in this paper:

1. We identify smart home usage scenarios particularly relevant for access control and use those to derive unique requirements for access control in future smart home platforms.
2. We use OpenHAB, a widely used smart home platform, as a case study and first analyze its current access control approach. Building on these rather lacking techniques, we then describe our efforts of retrofitting OpenHAB with access control features that are ready for future deployments in open, web-based environments.
3. We identify key lessons learned for both users of current as well as developers of future smart home platforms.

This paper is structured as follows: We describe relevant usage scenarios in smart home environments and identify core requirements for access control in such systems in Sect. 2. Afterwards, in Sect. 3, we give a brief overview of the OpenHAB platform, describe its current access control approach and our efforts in making said platform ready for open, web-based deployments. Building on this, we discuss our observations and lessons learned in Sect. 4 before concluding in Sect. 5.

2 Access Control in Smart Home Systems

In matters of access control, IoT Systems and, in particular, smart home platforms differ significantly from other systems in various respects, leading to specific requirements that have to be met. While “basic use cases” of smart home platforms are often perceived trivial in matters of access control – all members of a family living in a house have full control, for example – things change significantly when “non-regular” use cases and scenarios come into play. Some such scenarios shall thus be briefly outlined in the following.

2.1 Relevant Usage Scenarios

One scenario that any smart home platform will sooner or later have to address properly comprises guests staying within the covered premises [13]. This may include private visitors as well as hotel- or AirBnB-like settings. In any such case, guests should, for example, be able to control and program the behavior of their windows’ blinds as well as to monitor and control the temperature of their respective room during their stay but have no access to similar functions for other private rooms. Also, there might be a need to allow guests to read the status of shared devices like the temperature sensors in the hallway or to

dynamically connect the smart home platform with external services (for receiving personalized email notifications or traffic forecasts, for example). Besides the core functionality required to implement all these aspects, associated processes of user enrollment, provisioning of access rights, etc. should of course also be as uncomplicated and “frictionless” as possible.

Another common scenario refers to external trust persons legitimately sojourning in the covered area – e.g., cleaning or nursing staff or simply different friends of the inhabitants looking after plants. As such persons are commonly trusted to physically enter the premises and thus to have physical access to most devices located within (e.g., lamps, window blinds, heating, ...), it makes no sense to prevent them from having the same access in the digital domain. However, providing them with respective access on the basis of pre-known identities may often prove challenging: In the case of cleaning or nursing staff, for example, the actual persons legitimately present within the premises often differ unpredictably, depending on dynamic schedules and ad-hoc substitutions. Using their physical presence itself is therefore a more promising starting point for suitable access control mechanisms. Alternatively, an approach more in line with the process used for providing physical access itself – the cleaning company is trusted, thus provided with a key and authorized to hand over this key to its personnel – may also serve as blueprint for suitable access control here. A combination of several approaches may also be needed from time to time: for instance, staff may start out with limited privileges simply based on their employment relation and physical access to certain areas but may acquire additional rights over time based on increased trust at a personal level.

With increasing interconnection of things, services, and APIs, it also becomes more and more important to provide suitable access control to an open, specifically non-closed smart home platform. For instance, hitting a dash button to order some products also has monetary effects on the owner’s bank account – controlling who can invoke external APIs should not be disregarded. Also, while external services can be integrated through invocation and polling, sometimes such third party services will have to trigger events in the smart home system. For instance, one might define a set of rules on IFTTT or FRED [5] that analyzes current wind measurements and controls window blinds to protect them from permanent damage. Or one could use an external service that offers precisely that functionality. Last but not least, there is a number of separate home automation gadgets, e.g., Google Home or Amazon Echo, or even simply a smartphone. Integrating such devices requires the necessary access control logic to grant extensive but not unlimited rights to those devices. All in all, this means that fine-grained access control with intelligent learning and delegation logic must not be limited to human users but also needs to extend to external services, platforms, and APIs.

Finally, the core problem that strict and highly elaborate access control models often prevent dynamic reactions to unforeseen situations will in all likelihood become increasingly prominent in the smart home context. Assuming a scenario where a person in need of care does not respond to calls on the door, it might

be reasonable if, for example, police officers could request temporary access to an in-house camera or to the door lock from, e.g., the person's children living in another city. Alternatively, it might also be reasonable to allow certain parties like emergency services to "override" access restrictions in a controlled manner that can be thoroughly examined afterwards. In any case, "preparing for the unanticipated" will be of utmost importance for suitable and usable access control in smart homes – not only in cases of emergency.

2.2 Technical Requirements and Mechanisms

Based on the above scenarios, we have identified a number of core requirements for access control in web-enabled smart home platforms. In this Section, we present these requirements and briefly discuss mechanisms we believe to be useful for addressing them. All requirements and mechanisms identified below are intended to be non-exclusive, allowing for multi-condition access control combining multiple criteria from the same category as well as from different ones.

Frictionless Usage: Smart home scenarios pose a particular need for smoothness. Especially in the living context, smart technologies should not be perceived as separate "add-ons" but must rather be embedded or "woven into" [18] established behavioral patterns and habits. Guests – in a hotel as well as in a private context – should not feel bothered by a need to familiarize with smart components or by cumbersome setup or "enrollment" procedures. Necessary management efforts should also be low on the operator's side. This is particularly true for access control mechanisms, where repeated commissioning and decommissioning of users and fine-grained assignment of rights could easily raise significant overheads. For all our considerations below, ensuring frictionless usage is thus a core design goal. Furthermore, frictionless usage may also be supported by specifically targeted mechanisms of self-learning etc.⁵

Fine-Grained Access Control: One core requirement that can be identified from the above scenarios is the need for fine-grained access control. It must be possible to provide different parties with different access rights for monitoring and control on a per-resource (devices, software components and services) level. To lower the efforts necessary for specifying and managing respective access rights, device grouping should also be possible. In terms of concrete mechanisms, this requirement resembles functionality already provided by most modern access control schemes. With minor adaptations, established mechanisms for fine-grained access control should thus be transferable to the smart home domain at reasonable effort.

Temporary Permissions: As motivated by the guest-related scenario, temporariness of access rights plays a more prominent role in the smart home context than it typically does in other domains. Even though granted access rights might

⁵ Manifold further approaches could be thought of here. Kim et al. [13], for instance, also experimented with automated rights assignment based on social network and phone records analysis.

always be revoked manually, directly issuing them for a limited timeframe would thus be of particular value for smart home platforms, also serving the goal of frictionless usage. Technology-wise, access tokens with limited validity periods are widely used in practice for comparable purposes. These might be used in combination with pre-existing user accounts (e.g., when access rights have been temporarily escalated and activities must be attributable to the respective user) or as a completely isolated modality of authentication, depending on the intended use case.

Context-Driven Permissions: The existence and relevance of physical context is one of the main specifics distinguishing access control in smart homes from other application domains. In the smart home domain, any device as well as any user striving for access has a physical presence and context which can serve as a foundation for access control concepts specific to smart home scenarios. In the above-mentioned case of cleaning or nursing staff legitimately being in a house, for example, any evidence proving that a user actually is within certain premises may be deemed a sufficient basis for granting access to all devices located in these premises [13]. Sufficiently reliable proofs for being in a certain room might, for example, be based on ultrasonic sound [7] or indoor light [11] used to transfer secrets to the client device. Independently from the concrete mechanisms used, however, we see permissions based on physical context – a particular form of attribute-based access control [16, 19] – as an important component of suitable access control for smart home and IoT scenarios.

Rights Delegation: Scenarios like the one referring to cleaning or nursing staff above also motivate a need for rights delegation. Instead of access being granted based on physical access (which particularly proves impractical in the case that door locks are also managed by the access control system), rights delegation could also be used: The cleaning company would then be granted access to certain devices, combined with the possibility to “hand over” these access rights to its employees. Such capabilities for rights delegation may also serve the overall goal of frictionless usage and the need to “prepare for the unanticipated”, as it allows to dynamically align access rights to changing needs. Rights delegation can be implemented in various ways, ranging from role delegation [1] to certificate driven solutions [4]. With a particular focus on IoT, rights delegation can be implemented through capability-based delegation [3], attribute-based access control, or through adaptations of the OAuth protocol [9].

On-request Access Provisioning by Authorized Parties: Closely coupled with the basic need for rights delegation is the requirement of being able to explicitly request access. As laid out in the scenario with a non-responding person and as obviously given in many less dramatic situations, actual access needs in smart home environments can hardly be foreseen completely but rather often emerge and need to be fulfilled ad-hoc. Beyond pure rights delegation, well-defined mechanisms for requesting not yet existing access from authorized persons and for respective “on-request delegation”⁶ are thus an important

⁶ Elsewhere, this model is also called “ask for permission” [13].

building block of “preparing for the unanticipated”. In addition to those mechanisms already used for basic rights delegation, techniques for on-request delegation can build upon established concepts such as authentication proxies [15] or dynamic consent [12].

Break-Glass Overrides: Another approach of “preparing for the unanticipated” is the concept of break-glass access control. In this model, emergency access to resources (like, in the example above, the camera within a non-responding person’s flat) is possible as soon as the “breaking” party (e.g., a police officer) can be reliably identified and appropriate notifications about the act of breaking are sent out to trigger ex-post evaluations. Having their roots in healthcare and disaster management, break-glass override mechanisms are highly relevant whenever strict access restrictions could potentially lead to significant harm in unanticipated situations. They might thus be of particular relevance in the smart home context whenever reasonable emergency access would today be achieved by means of physical destruction, for example. Technically, break-glass mechanisms can be implemented on top of or natively integrated into existing access control mechanisms [6], whereas the ability to actually perform an emergency access might be given to anybody or limited to a well-defined set of parties. Blockchain technology might be helpful for implementing non-disputable break-glass logs.

3 Case Study: OpenHAB

In the following, we discuss openHAB, the IoT framework we used to prototypically implement some of the mechanisms described in Sect. 2.2. We first give a brief overview of openHAB, its architecture and how it handles access control in its default configuration. Afterwards, we delineate how we extended its access control mechanisms to better meet the requirements we developed in Sect. 2.

3.1 System Overview

As a case study, we chose the OpenHAB project, mainly because it represents a typical IoT eco system and unlike most others is open source. OpenHAB is an extension of the Eclipse SmartHome project and explicitly built for easy integration of new devices, protocols, and services. Most of its components interact through a pub/sub event stream, the “**Open Home Automation Bus**”.

The project also comes with a modern interface and one of the most exhaustive and fastest-growing repositories for device bindings and third party integrations on the market, which together led to a growing community of users.

The framework is based on Java and composed of a set of OSGi (Open Service Gateway Initiative) services and components. Its architecture can be broken down into three major component groups: (1) the core: including the message bus, configuration storage, logging; (2) binding provider: responsible for communicating with devices and services; (3) user interfaces and user services, including the rule engine employed to run user-defined code and the REST

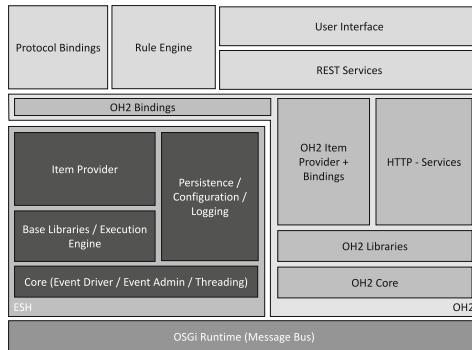


Fig. 1. Overview of the OpenHAB System Architecture. Showing the Eclipse SmartHome Core (left) and the OpenHAB specific extensions (right) as well as common modules.

interface that can be utilized by external services. Figure 1 shows a more detailed view of the OpenHAB architecture.

Any extension to OpenHAB is added as an OSGi binding, which can access every other service within the OpenHAB runtime including all data stored within the system without any constraints.

3.2 Access Control in OpenHAB

OpenHAB has a very basic access control system, which only allows a single user/password combination to be set. If set, the respective user becomes the only entity able to access any and all of the openHAB features. By default, however, setting a user/password combination is neither enforced nor at least suggested. Having only one single account clearly limits openHAB's usability for more complex use cases as presented in Sect. 2.1.

Furthermore, OpenHAB enforces access control only in the third group of components (user interfaces), while bindings and user-defined rules can access any internal service or any other resource on the local or public network without providing any means of authentication. Even though clearly fostering the easy development and addition of new types of services and system resources as OpenHAB components, such a full-access paradigm is obviously unsuitable for more complex settings with hundreds or thousands of controlled devices, with the platform being interconnected with multiple external services, and with external users potentially being able to connect their own devices and external services: Under the current concept, just one exploitable or unexpectedly behaving component could suffice to compromise or disturb the whole installation. Together with the risk-prone dynamic update or addition of components inherent to the OSGi concept, this sums up to a security concept and subsystem that is everything but “production-ready” beyond rather trivial use cases in closed networks. Finally, none of the advanced requirements identified in Sect. 2.2 is currently fulfilled by OpenHAB.

3.3 Extending Access Control

Given OpenHAB's above-mentioned shortcomings, we set out to fix its core security model and to prototypically implement mechanisms for meeting the requirements identified in Sect. 2.2. For doing so, we had to overhaul most parts of OpenHAB's existing access control mechanisms and retrofit our own. To ease management effort and heighten practical usability in non-trivial use cases, we added a way to group devices in multiple overlapping security groups, established a way to record the use of each device, and introduced a rudimentary user registry.

In order to concentrate on the internal processes of the access control system, we confined our development efforts to an installation with only the REST interface being active. With our extensions, all access control information is transmitted using a mechanism based on JSON web token (JWT) in a custom HTTP-header. For use cases involving attribute-based access control, a client can either compute these tokens (based on context information, for example) or listen for a local broadcast token. Each response from our system can also contain a new token for the client to use, like a generated identifier that is similar to an automatically generated username.

Implementing Fine-Grained Access Control: Based on the just discussed infrastructure functionality, we implemented fine-grained access control by exploiting the already existing metadata infrastructure of OpenHAB. It provides a set of tags for each device; we use these tags to identify relationships between devices. Once an entity can prove access to a device, the user might also be granted access to other devices with similar tags. We use the groups and device capabilities encoded in these tags to assess the risk of unauthorized access. In the absence of other authentication information, the calculated risk is used to make access decisions. To fully support fine-grained access control, we also do not enforce access control simply at the user interface but at the core components responsible for issuing commands to devices.

Implementing Rights Delegation: We implemented rights delegation mainly by relying on the above-mentioned user management; it holds rights that each user accumulates over time. Delegation then is handled by linking the delegated sets of rights to a new user (thus following a user-driven approach to prevent uncontrolled onward delegation). The original rights holder can still use these rights, revoke or modify delegations, and delegate the same rights to further users.

Implementing Temporary, Context-Driven, and On-request Permissions: We used attributes codified in tokens to determine if a user could access a resource. Each attribute must also be signed, which allows the system to ensure that an attribute could not be manipulated. These attribute tokens can encode location information as well as context information. Once a request is issued to the system, these attributes are evaluated by the respective controller classes. All attributes are evaluated individually for each device affected by the request. Depending on the evaluation outcome, it might be that the response only contains a filtered set of information. A controller might also decide only to perform

a subset of the actions that were requested. Using similar concepts, we also implemented on-request access provisioning, where the controller puts respective commands on hold until a privileged user has approved it.

Implementing Break-Glass Overrides: Finally, we added a break-glass override mechanism by allowing our access control system to be shutdown if the right set of keys is used. These keys are generated by a token generator that would have to be given to those parties that should be able to perform an override (the local fire department, a neighbor, etc.). The owner of the OpenHAB installation has total control over these generators and can disable them at any time. Once the break-glass override is invoked, all access control mechanisms are disabled, and only the audit component is still active. The notification system of the on-request delegation system is used to inform all privileged users what is happening and which token was used to invoke the breaking the glass override.

Means for Frictionless Use: To serve the overall goal of frictionless usage both on the user and the management side, our system can learn from each interaction and over time build a simplified model about what a specific user is allowed to do. Once multiple comparable requests (like “light X in room Y”) have repeatedly been approved, the user no longer needs to ask for permission for similar requests. This is done by recording each request, each attribute used for that request and the outcome of these requests. We use this information to train a simple risk model. If the risk factor for a given request is low enough, then the request will be successful regardless of the content of other provided attributes. We also record these automatic approvals for a later audit.

Specific Challenges of OpenHAB: Intuitively, implementing these features should have been comparably simple. However, OpenHAB’s architecture made it rather difficult. In particular, the OSGi framework, which openHAB uses to dynamically load and replace processes at runtime, could have been used to disable any feature our access control add-on would provide. To get around this problem, we enhanced the control flow of information between all components responsible for executing commands (e.g., ItemProvider and Execution Engine). We did this by implementing an auditing framework that analyzed the stack trace of each incoming method call and only allows a predefined set of the classes to execute OpenHAB commands. Any OSGi service can still see all internal services, but if a non-listed service tries to create a request, all commands are dropped.

The OSGi framework also leads to another problem regarding the communication between our access control implementation and other components in the system. The original design of openHAB did not allow for session information to be transported to lower-level components. This meant that we could not filter communication between different layers on a per-resource basis. We, therefore, had to build a complicated workaround to transport simple session information between classes: Given that we could not change methods inside the architecture without rewriting all OpenHAB device bindings, we had to transfer information indirectly. In particular, we created an OSGi service that interacted with the

Java Debug Interface. This service allowed us to correlate session information with stack traces and method calls which allowed us to evaluate device interactions on an event by event basis. In matters of performance, maintainability, and transparency, this approach is, however, by no means ideal.

Besides these limitations, we were able to add the on-request-delegation system, location broadcasts and other extensions mentioned in Sect. 2.2 without any hindrance. Besides implementing the user interfaces for these services, we also added a set of cryptographic utilities that are, for instance, used to generate the time-dependent tokens using a HMAC-based one-time password algorithm.

4 Discussion

The lesson learned no.1 is that retrofitting access control is a rather bad idea; instead, it needs to be considered as a core design goal from the very beginning especially for an open, web-connected platform that integrates external services and data sources with a non-fixed user base. It is also important to handle access control not only on an exterior interface but rather internally on a component or device level. With OpenHAB's current overall system architecture, however, applying the presented techniques from Sect. 2.2 or even making the slightest extensions to the existing security design is virtually impossible even though we managed to achieve a working solution. However, considering that complexity is the worst enemy of security, the necessary workarounds and indirections for retrofitting advanced access control that we took cannot be recommended safely.

While we only analyzed OpenHAB in such detail, access control does not seem to be a core design goal in other platforms either. For addressing advanced scenarios as described in Sect. 2.1 and being web-ready including interaction with third party services, smart home platform developers should probably rethink their priorities. In this context, technology such as OSGi also has both benefits and disadvantages – deciding on such technology stacks should be the result of careful deliberation.

Finally, an important consideration is also to design access control in a way that is compatible with widely used, established web standards such as OAuth or even WS-Security. Otherwise, all integration of external services, data sources, and APIs is ripe for malicious exploitation and not advisable.

5 Conclusion and Future Work

The Internet of Things has been finding more and more adoption in the last few years and is seeing continuous growth; particularly, smart home technology has found more and more adoption. Smart home platforms, however, still mainly run in isolated networks not connected to the public Internet. We expect this to gradually change towards more open deployments including external services on the web and in the cloud. In such deployments, however, access control based on mechanisms specifically tailored to the smart home domain is of utmost importance.

In this paper, we started by discussing unique use cases in future smart home platforms and used these to derive specific requirements for access control. Afterwards, as a case study, we analyzed OpenHAB (as a popular example of a smart home platform) in detail with regards to its access control mechanisms. Since we found these rather lacking, we then tried to retrofit state-of-the-art access control in compliance with our identified requirements. Through this case study, we learned that at least one of the current smart home platforms (and potentially even other more general IoT platforms) is not ready for future, more complex application scenarios. We also identified how smart home environments offer unique ways for access control based on physical context. Finally, we learned that retrofitting access control in existing smart home platforms is painfully complicated and cannot be recommended – both from a security and an effort perspective.

In our future work, we plan to test other IoT Hub platforms and see if some of the limitations we observed can be improved by using a more suitable platform where fewer indirections and workarounds are needed to integrate our fine-grained access control method. Furthermore, we intend to test our methods not only on edge devices but also expand our access control approach to the dispersed and interconnected fabric of fog computation.

Acknowledgments. This work partly been supported by the European Commission through the Horizon 2020 Research and Innovation program under contract 731945 (DITAS project).

References

1. Ahn, G.J., Mohan, B.: Secure information sharing using role-based delegation. In: International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004, vol. 2, pp. 810–815 (2004)
2. Andy Greenberg: Hackers Remotely Kill a Jeep on the Highway—With Me in It. <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>
3. Anggorojati, B., Mahalle, P.N., Prasad, N.R., Prasad, R.: Capability-based access control delegation model on the federated IoT network. In: 2012 15th International Symposium on Wireless Personal Multimedia Communications (WPMC), pp. 604–608 (2012)
4. Aura, T.: Distributed access-rights management with delegation certificates. In: Vitek, J., Jensen, C.D. (eds.) Secure Internet Programming. LNCS, vol. 1603, pp. 211–235. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48749-2_9
5. Blackstock, M., Lea, R.: Fred: a hosted data flow platform for the IOT built using node-red. In Proceedings of MoTA 2016 (2016)
6. Brucker, A.D., Petritsch, H.: Extending access control models with break-glass. In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. pp. 197–206. ACM (2009)
7. Chen, K., Aljarrah, M., Bonnet, P.: Leveraging physical locality to integrate Smart appliances in non-residential buildings with ultrasound and Bluetooth Low energy. In: Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016 1(iii), pp. 199–209 (2016)

8. Dong, R., Ratliff, L.J.: Privacy in the Internet of Things. *Next Wave* **21**(2), 8–16 (2016)
9. Emerson, S., Choi, Y.K., Hwang, D.Y., Kim, K.S., Kim, K.H.: An oauth based authentication mechanism for IoT networks. In: International Conference on ICT Convergence 2015: Innovations Toward the IoT, 5G, and Smart Media Era, ICTC 2015, pp. 1072–1074 (2015)
10. Gartner: Gartner Says 8.4 Billion Connected “Things” Will Be in Use in 2017, Up 31 Percent From 2016 (2017). <http://www.gartner.com/newsroom/id/3598917>
11. Grobe, L., Paraskevopoulos, A.: High-speed visible light communication systems. *IEEE Commun. Mag.* **51**(12), 60–66 (2013)
12. Kaye, J., Whitley, E.A., Lund, D., Morrison, M., Teare, H., Melham, K.: Dynamic consent: a patient interface for twenty-first century research networks. *Eur. J. Hum. Genet.* **23**(2), 141–146 (2015)
13. Kim, T.H.J., Bauer, L., Newsome, J., Perrig, A., Walker, J.: Access right assignment mechanisms for secure home networks. *J. Commun. Netw.* **13**(2), 175–186 (2011)
14. Liu, J., Xiao, Y., Chen, C.P.: Authentication and access control in the Internet of Things. In: Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems Workshops 2012, pp. 588–592 (2012)
15. Mayrhofer, R.: A context authentication proxy for IPsec using spatial reference. In: Proceedings of TwUC 2006: 1st International Workshop on Trustworthy Ubiquitous Computing, pp. 449–462. Austrian Computer Society (OCG), December 2006
16. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in Internet of Things: the road ahead. *Comput. Netw.* **76**, 146–164 (2015)
17. Vallance, C.: Car hack uses digital-radio broadcasts to seize control (2015). <http://www.bbc.com/news/technology-33622298>
18. Weiser, M.: The computer for the twenty-first century. *Sci. Am.* **265**, 94–100 (1991)
19. Yuan, E., Tong, J.: Attributed Based Access Control (ABAC) for web services. In: Proceedings - 2005 IEEE International Conference on Web Services, ICWS 2005, pp. 561–569 (2005)



Integrating Smart Devices as Business Process Resources – Concept and Software Prototype

Robert Wehlitz^{1(✉)}, Ingo Rößner¹, and Bogdan Franczyk^{2,3}

¹ Institute for Applied Informatics (InfAI), Hainstr. 11, 04109 Leipzig, Germany
{wehlitz, roessner}@infai.org

² Information Systems Institute, Leipzig University, Grimmaische Str. 12, 04109 Leipzig, Germany
franczyk@wifa.uni-leipzig.de

³ Business Informatics Institute, Wrocław University of Economics, ul. Komandorska 118-120, 53-345 Wrocław, Poland

Abstract. The foundation of the Internet of Things (IoT) consists of ubiquitous smart devices, equipped with sensors, actuators and tags, that are connected to the Internet and able to communicate with one another. It is seen as a great opportunity for organizations to improve, innovate, and reinvent their business processes. However, existing Business Process Management (BPM) tools and systems are not fully capable of integrating and utilizing smart devices as business process resources and, thus, are unable to unlock the high joint potential of BPM and the IoT. In this paper, we propose a concept for a service-oriented BPM system architecture which aims at the modeling, implementation, and execution of IoT-aware business processes. Furthermore, we introduce a first software prototype of the architecture as a proof of concept.

Keywords: Business Process Management systems · Internet of Things · Service-oriented architectures

1 Introduction

Business Process Management (BPM) is nowadays a well-recognized discipline in academia and practice in order to define, improve, and manage business processes (BP). In the last decades, a plethora of BPM tools and systems, which enable process-oriented Human-to-Human, Human-to-System, and System-to-System interaction, has emerged [1]. With the advent of the Internet of Things (IoT), where physical world objects are always connected to the Internet through ubiquitous smart devices, novel interactions during BP, e.g. Human-to-Thing or Thing-to-Thing, are possible [2]. However, current BPM systems are not fully capable of integrating and utilizing smart devices as BP resources [3].

In light of this, the main contribution of this paper is a concept for a service-oriented IoT-aware BPM system architecture which enables to model, implement, and execute BP that use sensing and actuation capabilities of smart devices in smart building as well as smart home environments. Furthermore, we present a software prototype as a proof of concept. The paper is structured as follows: First, we

briefly discuss some aspects on how both businesses and customers could benefit from BPM in conjunction with the IoT and touch upon related work (Sect. 2). We then present our concept on how to integrate and utilize smart devices as BP resources (Sect. 3). The corresponding software prototype is introduced in (Sect. 4). The paper concludes with a short summary and an outlook for future research (Sect. 5).

2 Motivation and Related Work

The total number of smart devices connected to the Internet is estimated to increase to around 24 billion (without smartphones and tablets) in 2020 [4]. This will lead to massive amounts of data which can be utilized by businesses to improve their BP [5]. For instance, high-frequently recorded sensor data from smart devices can serve as a basis for faster and improved decision making during BP execution, e.g. the early detection and handling of delivery problems in logistics networks through detailed package tracking. Furthermore, performance weaknesses and bottlenecks of BP can be identified resp. prevented, e.g. using predictive maintenance techniques in order to avoid cost-intensive unscheduled downtimes in production systems.

In the context of smart building and smart home environments, customers may benefit from new products and services in the areas of entertainment, convenience, security, health care, and energy efficiency [6]. Businesses, in turn, have the possibility to gather and analyze more detailed data about their customers in order to individualize and improve their products and services [2].

All of this requires that services, data, and events are appropriately orchestrated among businesses, smart devices, and customers. A task BPM systems seem to be well suited for [3]. However, there are many challenges current BPM systems are not fully addressing. For example, the heterogeneity of smart devices which includes different hardware specifications, communication protocols, data exchange formats, and discovery mechanisms [7]. As a first step towards holistic IoT-aware BPM systems, we found it fundamental to provide a system architecture which is able to integrate different smart devices as BP resources, hence, overcoming the heterogeneity issue.

Research in this field is still at its beginning. A literature survey conducted by [7] revealed that numerous publications focus on modeling IoT-aware BP without discussing technical details concerning the integration of smart devices. They also provide no further information on the implementation and execution of IoT-aware BP. Some solutions take into account implementation aspects; however, only concentrate on the integration of sensors or tags, which do not provide actuation capabilities. Other research work depends on the presence of gateway topologies or additional hardware required on the local network site which prevents 1:1 communication between, for instance, mobile devices and the BPM system. Furthermore, most solutions lack a monitoring component that provides transparency on BP execution. For this reason, we designed a BPM system architecture that enables the integration of smart devices for modeling, implementing, and executing IoT-aware BP.

3 Concept

In order to be able to use sensing and actuation capabilities of smart devices, we describe them at type and instance level. A device type (e.g. *Temperature Sensor*) represents a set of similar device instances. Device instances, in turn, are concrete occurrences of the respective device type (e.g. *Temperature Sensor #1*, *Temperature Sensor #2*, etc.). Each device type provides one or a set of service types (e.g. *getTemperature*) which may require to specify input, output, and configuration parameters (e.g. *interval*, *unit*). Device instances provide service instances (e.g. *getTemperature* from *Temperature Sensor #1*) whose parameters have concrete values (e.g. *interval: 10, unit: seconds*). Similar to smart devices, rules to identify events in data streams can also be described at type (e.g. value >22) and instance level (e.g. *temperature value from Temperature Sensor #1 > 22 °C*).

In the following, we introduce our first concept for a service-oriented BPM system architecture (see Fig. 1) that we derived from literature analysis, internal workshops, and software prototyping. It supports the modeling, implementation, and execution of IoT-aware BP and is based on a centralized orchestration model [7].

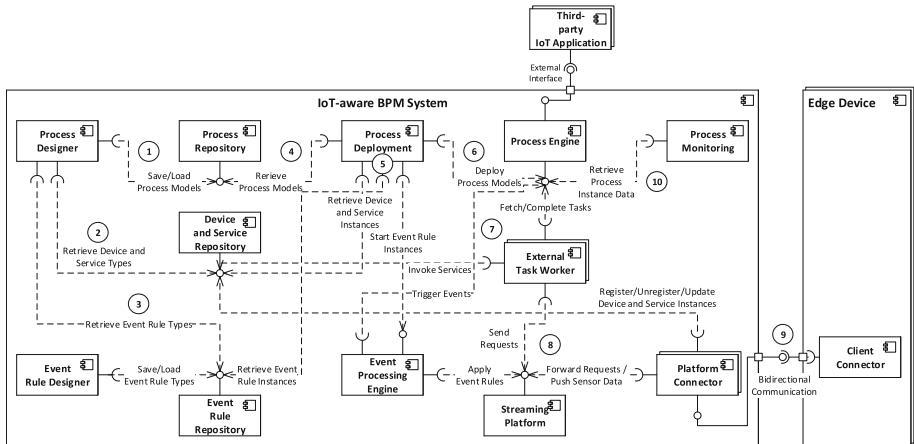


Fig. 1. UML component diagram of the IoT-aware BPM system

A *process designer* component is provided for the graphical modeling of BP. The BP models are stored in the *process repository* (①). During BP modeling, only types of events, smart devices, and device services are available for use. This allows for the instance-independent composition and reuse of BP models at design time. Furthermore, it enables to deploy the same BP model multiple times for different smart devices. The *process designer* fetches device-related metadata (device and service types) from the *device and service repository* (②) which enables the unified access on heterogeneous devices by service abstraction. Event rule types, which represent event types and are defined by means of an *event rule designer*, are stored and retrieved from the *event rule repository* (③). This approach allows for event, device, and service types to be directly referenced by BP elements during modeling.

The *process deployment* component fetches executable BP models from the *process repository* (④) and examines if they contain event and/or device and service types. If this is true, the *process deployment* component retrieves eligible instances of all device and service types used in a BP model from the *device and service repository* (⑤). It then lets the user decide which specific device and service instances shall be utilized for a new BP deployment. The same applies to event types. The user defines which generic event rule types shall be applied on specific data streams of smart devices. Thereafter, the BP model is enriched with instance-related metadata and deployed to the *process engine* (⑥), which ensures that BP instances are executed according to the underlying model.

The BP service task execution is done by *external task workers*. They fetch pending jobs from the *process engine* and inform it as soon as the jobs have been completed so that it can proceed with the next BP step (⑦). *External task workers* invoke sensing and actuation services (e.g. *getTemperature* or *turnOffLight*) by using metadata of event, device, and service instances which is embedded in the deployed BP definitions. In case of executing an actuation service, an *external task worker* fetches the required service description from the *device and service repository* and sends a service request to the *streaming platform* (⑧). The *streaming platform* provides an interface by which *platform connectors*, that are responsible for handling the communication between the IoT-aware BPM system and edge devices, can fetch pending service requests and forward them to their counterpart in local networks, that is *client connectors* (⑨). *Client connectors* run on smart devices themselves or on gateway devices and represent one of the key components for overcoming the heterogeneity issue mentioned in Sect. 2. They are responsible for the registration and (auto-)discovery of smart devices and pass sensing data from local networks on to the IoT-aware BPM system. Furthermore, they forward requests to the service endpoints of smart devices. The *platform connectors* push incoming sensing data to the *streaming platform* where it can be filtered by events through the *event processing engine*. The *event processing engine* applies event rules on data streams and triggers events if corresponding rules are met. These events can be processed by the *process engine*, e.g. to start a new or to affect the control flow of a running BP instance. Finally, a *process monitoring* component accesses the interface of the *process engine* in order to obtain historical and operational data on BP instances (⑩). This includes, for example, the number of completed or running instances of a certain BP model, throughput times, or errors that might occur during BP execution.

4 Software Prototype

In order to provide a proof of concept of our architecture presented in Sect. 3, we implemented all system components (see Fig. 1) as an integrated cloud-based software prototype. This is part of an IoT platform that aims at the provisioning of tools which support the development of digital services in the area of home energy management.

The *process designer* component of our software prototype is based on bpmn.io, a BPMN 2.0 rendering toolkit and web modeler [8], which was extended to integrate with the *device and service repository*. We selected BPMN 2.0 as supported modeling

language because it seems to be the best suited standard for mapping IoT concepts in BP [9]. Following [10], our solution intends that IoT-aware BP are modeled within BPMN pools. The lanes of a pool represent device types. Each service task that is placed on a lane is assigned to and handled by the respective device type. The user clicks on a service task and selects a supported service type from a dialog window. The label of the service task in question is automatically updated with the device-related metadata retrieved from the *device and service repository* to enhance the comprehensibility of the BP diagram.

In case of a new BP deployment, the *process deployment* component analyzes the BP definition and prompts the user to select concrete instances for every found event and/or device type. It then deploys the BP model to the *process engine*, which is based on the open-source platform Camunda [11]. This supports the external task pattern and, hence, provides more flexibility and scalability for executing BP service tasks [12]. The implementation of *external task workers* is independent from a specific programming language and additional source code for BP service task execution has not to be deployed to and executed by the *process engine*.

Most of the smart devices we use as BP resources for testing our approach support the wireless communication standards ZigBee or Z-Wave. Both are very popular and well suited for home energy management scenarios [13]. The smart devices are connected to local ZigBee or Z-Wave networks as well as to the Internet via gateway devices. We implemented lightweight *client connectors* and deployed them on the gateway devices. A software library, written in Python, is also available in order to integrate “things” which are not part of gateway topologies. Service requests from *platform connectors*, which address service endpoints of smart devices, are sent to *client connectors* over WebSocket connections. This allows the communication with local networks from the Internet without the need for cumbersome router configuration. With regard to the streaming platform, we use Apache Kafka and its publish-subscribe messaging system [14]. *External task workers* operate as producers and publish service requests to Kafka topics to which *platform connectors* are subscribed. The *platform connectors* are also producers and publish sensing data to topics which, in turn, can be subscribed by the *event processing engine*. This enables a high flexible and scalable handling of incoming resp. outgoing event and data streams.

5 Conclusion and Outlook

In this paper, we discussed potential use cases for BPM in conjunction with the IoT and outlined related work as well as the need for other research contributions. Furthermore, we presented a concept for a service-oriented BPM system architecture which supports the integration of smart devices as BP resources and introduced a first implementation as a proof of concept. In this context, we showed how we tackle the heterogeneity challenge by using connectors and metadata stored in the *device and service repository* for service abstraction in order to provide unified access to smart devices. Furthermore, system components which make use of these metadata and support the modeling, implementation, and execution of IoT-aware BP were described.

However, the results presented in this paper are work-in-progress and provide a basis for future research. The existing concept must be developed further towards a hybrid architecture which allows centralized as well as decentralized BP execution in order to tackle other challenges, such as privacy, security, and quality of service. In addition, tools and techniques for adaptive case management should be investigated in order to improve context awareness of process-oriented IoT applications. This will shift the utilization of BP from well-structured workflows to adaptive cases whose control flow is determined at run time based on context information.

Acknowledgments. The work presented in this paper is partly funded by the European Regional Development Fund (ERDF) and the Free State of Saxony (Sächsische Aufbaubank – SAB).

References

1. van der Aalst, W.M.P.: Business Process Management. a Comprehensive Survey. ISRN Software Engineering, 2013 (1), 37 p. (2013)
2. Khoshafian, S., Schuerman, D.: Process of everything. In: Fischer, L. (ed.) iBPMS. Intelligent BPM Systems – Impact and Opportunity, pp. 67–82. Future Strategies Inc., Lighthouse Point (2013)
3. Whibley, P.: The Internet of Things will be invisible. In: Fischer, L. (ed.) BPM Everywhere. Internet of Things – Process of Everything, pp. 39–46. Future Strategies Inc., Lighthouse Point (2015)
4. Greenough, J.: How the “Internet of Things” will impact consumers, businesses, and governments in 2016 and beyond (2016). Accessed 08 Aug 2017. <http://www.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?IR=T>
5. Francis, S., Gibbs, L.: I, For One, welcome our new robot overlords. In: Fischer, L. (ed.) BPM Everywhere. Internet of Things – Process of Everything, pp. 31–37. Future Strategies Inc., Lighthouse Point (2015)
6. Zion Market Research. Accessed 08 Aug 2017. <https://www.zionmarketresearch.com/news/smart-home-market>
7. Chang, C., Srivama, S.N., Buyya, R.: Mobile cloud business process management system for the Internet of Things. ACM Comput. Surv. **49**(4), 1–42 (2016)
8. bpmn.io Homepage. Accessed 09 Aug 2017. <https://bpmn.io>
9. Meyer, S., Sperner, K., Magerkurth, C., Pasquier, J.: Towards modeling real-world aware business processes. In: Proceedings of the Second International Workshop on Web of Things, pp. 1–6. ACM, San Francisco (2011)
10. Meyer, S., Ruppen, A., Magerkurth, C.: Internet of Things-Aware process modeling: integrating IoT devices as business process resources. In: Salinesi, C., Norrie, Moira C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 84–98. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_6
11. Camunda Homepage. Accessed 09 Aug 2017. <https://camunda.org>
12. Camunda Documentation. Accessed 09 Aug 2017. <https://docs.camunda.org/manual/7.7/user-guide/process-engine/external-tasks/>
13. Zhen, Z., Agbossou, K., Cardenas, A.: Connectivity for home energy management applications. In: Power and Energy Engineering Conference (APPEEC), pp. 2175–2180. IEEE PES Asia-Pacific, Xi'an (2016)
14. Kafka Homepage. Accessed 09 Aug 2017. <https://kafka.apache.org>



Architecting Enterprise Applications for the Cloud: The Unicorn Universe Cloud Framework

Marek Beranek¹, Marek Stastny¹, Vladimir Kovar¹,
and George Feuerlicht^{2(✉)}

¹ Unicorn College, V Kapslovně 2767/2, 130 00 Prague 3, Czech Republic

marek.beranek@unicorncollege.cz,

marek.stastny@unicornuniverse.eu,

vladimir.kovar@unicorn.eu

² Prague University of Economics,

W. Churchill Square. 4, 130 67 Prague 3, Czech Republic

george.feuerlicht@gmail.com

Abstract. Recent IT advances that include extensive use of mobile and IoT devices and wide adoption of cloud computing are creating a situation where existing architectures and software development frameworks no longer fully support the requirements of modern enterprise application. Furthermore, the separation of software development and operations is no longer practicable in this environment characterized by fast delivery and automated release and deployment of applications. This rapidly evolving situation requires new frameworks that support the DevOps approach and facilitate continuous delivery of cloud-based applications using micro-services and container-based technologies allowing rapid incremental deployment of application components. It is also becoming clear that the management of large-scale container-based environments has its own challenges. In this paper, we first discuss the challenges that developers of enterprise applications face today and then describe the Unicorn cloud framework (uuCloud) designed to support the development and deployment of cloud-based applications that incorporate mobile and IoT devices. We use a doctor surgery reservation application “Lekar” case study to illustrate how uuCloud is used to implement a large-scale cloud-based application.

Keywords: Cloud computing · Software frameworks · Micro-services
DevOps

1 Introduction

The basic enterprise computing objectives that include low cost, reliability, security, scalability, easy deployment and usability have not changed dramatically for decades. However, the hardware and software environment in which enterprise applications are being developed and deployed today is significantly different from that only a few years ago. Enterprise architectures have been evolving to take advantage of the

opportunities offered by advances in hardware and software technologies, in particular the increase in processing power and storage capacity and the corresponding cost reductions. The centralized architectures of the 1970s were superseded by client/server architectures of the 1980s, and component-based architectures of the 1990s, and most recently by the Service-Oriented Architecture (SOA) at the beginning of this century. We are now experiencing another major transformation driven by wide adoption of cloud computing and extensive use of mobile and IoT (Internet of Things) devices. Public cloud platforms (e.g. AWS [1], Microsoft Azure [2], etc.) offer highly elastic and practically unlimited computing power and storage capacity, allowing more flexible acquisition of IT resources in the form of cloud services, overcoming the limitations of on-premise IT solutions. Importantly, this new technology environment is creating opportunities for innovative solutions at a fraction of the cost of traditional enterprise applications. However, to take full advantage of these developments, organizations involved in the development of enterprise applications must adopt a suitable enterprise architecture and application development frameworks. The architecture needs to support various types of mobile devices (smart phones, tablets, etc.) and incorporate interfaces that interact with IoT devices. Unlike traditional enterprise applications that store data on local servers within the organization, most mobile applications store data and deploy application components in the cloud, making it possible for applications to be shared by very large user populations. Given the requirements of modern business environments, the architecture needs to facilitate rapid incremental development of application components, secure access to information and easy and fast cloud deployment. There is now increasing empirical evidence that to effectively address such requirements, the architecture needs to support micro-services and container-based virtualization [3]. However, it is also becoming clear that the management of large-scale container-based environments has its own challenges and requires automation of application deployment, auto-scaling and control of resource usage. The need for continuous delivery and monitoring of application components impacts on the structure and skills profile of IT teams, favoring small cross-functional teams leading to the convergence of development and operations (DevOps). The separation of code development and declarative methods of environment configuration play an important role in increasing the productivity of the software development process. Furthermore, developers of enterprise applications are increasingly turning towards open source solutions that allow full control over the entire software stack, avoiding costly proprietary solutions. Also, while the use of public cloud platforms is economically compelling, an important function of the architecture is to ensure independence from individual cloud providers, avoiding a provider *lock-in*. Finally, the architecture should reduce the complexity of the application development and maintenance process and facilitate effective reuse of application components and infrastructure services.

These requirements demand a revision of existing architectural principles and application development methods. In this paper, we describe the Unicorn Universe Cloud Framework (uuCloud) designed to facilitate the management of modern container-based cloud environments addressing the issues identified above. The uuCloud framework is an integral part of the Unicorn Application Framework (UAF). We have described the features of the UAF in an earlier publication [4], giving a

high-level overview of the architecture. In this paper, we focus on the uuCloud framework and describe its key components and operation. In the next section (Sect. 2) we review related literature focusing on DevOps, micro-services and container management frameworks. In the following sections, we briefly overview the UAF architecture (Sect. 3) and describe the uuCloud framework (Sect. 4). Section 5 illustrates the application of the uuCloud framework using an example of the Doctor’s Surgery Reservation System (Lekar Reservation System - LRS). Section 6 includes our conclusions and directions for future work. Please, note that all diagrams in this paper are drawn using the Unicorn Business Modeling Language: <https://unicornuniverse.eu/en/uubml.html>.

2 Related Work

Cloud-based application development frameworks and architectures have been the subject of intense recent interest by industry practitioners and academic researchers, in particular in the context of micro-services and DevOps [5]. As noted in Sect. 1, several recent trends including cloud computing, extensive use of mobile and IoT devices have impacted on the architecture of enterprise applications with corresponding impact on application development frameworks [6]. Namiot et al. [7] discuss the advantages and drawbacks of micro-services. The benefits of micro-services include the ability to use different programming languages for individual services, improved scalability and more rapid, incremental development within smaller teams. However, they also note that the use of a larger number of smaller services increases deployment complexity. One of the most important challenges involves decisions about how to partition the application system into micro-services, i.e. making decisions about service granularity. The paper discusses various service partitioning methods including the Scale Cube [8], partitioning by use-cases and partitioning by resource type. Armin Balalaie et al. [3, 9] consider achieving reusability, decentralized data governance, automated deployment and built-in scalability to be the main motivations for migration to a micro-services architecture. The authors report on their experiences during incremental migration and architectural refactoring of a commercial “Mobile Backend as a Service” to micro-services. They argue that standard virtualization methods introduce a heavy computational overhead and therefore are not cost-effective, and recommend the use of container-based virtualization to reduce overheads and to enable portability. The authors discuss synergies between micro-services and DevOps approach that involves small vertically structured cross-functional teams responsible for individual application components and services. They conclude that the main lessons learned from the migration to micro-services include: (1) the critical importance of service contracts as the number of services increases, (2) the need for skilled developers who understand distributed systems development, and (3) use of development templates. According to Rimal et al. [10] the most important current challenge is the lack of a standard architectural approach for cloud computing. The authors explore and classify architectural characteristics of cloud computing and identify several architectural features that play a major role in the adoption of cloud computing. The paper provides guidelines for software architects for developing cloud architectures. According to Raj

et al. [11] “The urgent thing is to embark on modernizing and refining the currently used application development processes and practices in order to make cloud-based software engineering simpler, successful, and sustainable.” The authors argue that software development has become an inherently complicated task and that a systematic, disciplined, and quantifiable approach is essential to make software development more manageable and to produce quality software products. A new requirements engineering process and techniques for capturing requirements for cloud-based services was proposed and illustrated using a large-scale case study based on Amazon Cloud EC2 [12]. Adaptation of the software development life cycle for cloud computing has been the subject of recent research interest. The differences between cloud service provider and consumer SDLC life-cycles resulting from the use of externally provided cloud services have been identified [13]. The authors describe a Service Consumer Framework (SCF) that incorporates architectural extensions designed to support operation in cloud computing environments [14].

While container technologies and micro-services have revolutionized application development and deployment, it is also evident that the use of these technologies has its limitations. More specifically, the management of large-scale container-based environments requires automation to ensure fast and predictable application deployment, auto-scaling and control of resource usage. At the same time, there is a requirement for portability across different public and private clouds. To address such issues a number of open source projects have been recently initiated; prominent examples include Cloud Foundry [15], OpenShift [16] and Kubernetes [17]. These projects share many common concepts and in some cases technologies. A key idea of these open source platforms is to abstract the complexity of the underlying cloud infrastructure and present a well-designed API (Application Programming Interface) that simplifies the management of container-based cloud environments. The Kubernetes project initiated by Google in 2014 as an open source cluster manager for Docker has its origins in an earlier Google container management system called Borg [18]. The Kubernetes project is hosted by the Cloud Native Computing Foundation (CNCF) [19] that has a mission “to create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self-healing multi-tenant nodes”. The objective is to facilitate *cloud native* systems that run applications and processes in isolated units of application deployment (i.e. software containers). Containers implement micro-services which are dynamically managed to maximize resource utilization and minimize the costs associated with maintenance and operations. CNCF promotes well-defined APIs as the main mechanism for ensuring extensibility and portability. A basic Kubernetes building block is a *Pod* - a REST object that encapsulates a set of logically connected application containers with storage resources (Volumes) and a unique IP address. Pods constitute a unit of deployment (and a unit of failure) and are deployed to *Nodes* (physical or logical machines). Lifetime of a Volume is the same as the lifetime of the enclosing Pod allowing restart of individual containers without the loss of data. Pods are externalized as *Services*; Kubernetes service is an abstraction that defines a logical set of Pods and a policy for accessing the Pods (i.e. micro-service). *Replication Controller* is used to create replica Pods to match the demand of the application and provide auto-scaling. Kubernetes uses *Namespaces* to partition resources allocated to different groups of users.

The application of Docker and Kubernetes container architecture to multi-tenant SaaS (Software as a Service) applications has been investigated and assessed using SWOT (Strength, Weakness, Opportunities and Threats) analysis and contrasted with developing SaaS applications using middleware services [20]. The authors conclude that more research is needed to understand the true potential and risks associated with container orchestration platforms such as Kubernetes.

While Kubernetes appears to be gaining momentum at present with support for major public cloud platforms including Google Cloud Platform, Microsoft Azure and most recently AWS, there is a number of other projects that aim to address the need for a universal framework for the development and deployment of cloud applications, including the Unicorn Universe Cloud framework described in this paper. While this rapidly evolving area is of active research interest to both academia and industry practitioners, currently there is a lack of agreement about a standard application development framework designed specifically for cloud development and deployment. Moreover, some proposals lack empirical verification using large-scale real-life applications.

3 Unicorn Application Framework (UAF)

The Unicorn Application Framework (UAF) developed by Unicorn (<https://unicorn.com/>) supports the design, development and operation of enterprise applications. A key UAF architectural objective is to support various types of mobile and IoT devices and to facilitate cloud deployment of enterprise applications utilizing standard framework services that include security and authentication services and support for multi-language environments. This minimizes the programming effort, improves reliability of applications and allows application developers to focus on the functionality that directly supports business processes and adds value for the end users. UAF architecture consists of four frameworks: uuUserInterface (uu5) - framework services for the development of Graphical User Interfaces (GUIs) based on HTML5 [21], uuIoT - framework services for the management of IoT devices, uuCloud - framework services for provisioning of elastic cloud services and uuAppServerKit - framework services for the development of application components (i.e. REST micro-services).

3.1 Unicorn Universe Application (uuApp)

UAF provides environment for the implementation and deployment of uuApp applications. uuApp is a component that implements a cohesive set of application functions designed to solve a set of specific user requirements. uuApp application is composed of sub-applications (uuSubApp) - independent units of functionality that implement specific business functions (e.g. booking a visit to a doctor's surgery). Each sub-application is implemented as a logical application server (uuAppServer) and is typically associated with a structured (uuAppObjectStore) or a binary (uuAppBinaryStore) data store. We made an architectural decision to associate each sub-application with a single *logical* application server to ensure fast access to persistent data and to maintain security and consistency of the underlying data sources. Using this approach, the access to underlying

data sources is controlled by the application server, ensuring that only authorized users can access the data. To improve scalability, individual use-cases (business functions) may be distributed across separate application servers in the form of individually addressable SPPs (Separately Performing Parts) modules.

The UAF follows the View-Model-Controller pattern, implementing the Model component in the form of an application server (uuAppServer) and the Controller and View components of the application on the client (typically a mobile device) using the uuUserInterface framework. The application server implements application logic and externalizes an API that is accessed by application clients. Persistent objects that belong to a sub-application are grouped into application *workspaces* (uuAppWorkspaces) and identified by an Application Workspace Identifier (AWID). Each sub-application is typically assigned a separate application workspace.

4 Unicorn Universe Cloud (uuCloud)

Unicorn Universe Cloud is a framework that supports autonomic provisioning of elastic cloud services using virtual containers and servers. UAF applications are typically deployed into a hybrid cloud environment (i.e. a combination of public and private cloud) in the form of virtualized application servers. To ensure portability and to reduce overheads, the UAF uses container-based virtualization. Our preferred containerization solution is Docker [22]. Docker container virtualizes the application including a complete filesystem that contains all components needed to run the application (i.e. system tools, system libraries, etc.) ensuring that the application runs independently of the platform the container is deployed on. A sub-application is mapped to an application server which is then containerized and deployed to a virtual server, and finally to a physical server. Docker containers can be deployed to a public cloud infrastructure (e.g. AWS or Microsoft Azure) or to a private (on-premise) infrastructure (e.g. the Unicorn platform Plus4U). Using containers for virtualization also improves isolation in multi-tenant environments [3].

4.1 uuCloud Nodes

We use a generic, technology agnostic terminology, *node* instead of container to indicate that applications can be implemented using containers or virtual and physical servers. Node is a unit of deployment with hardware characteristics that include virtual CPU (vCPU) count, RAM size, ephemeral storage, etc. Nodes are classified according to *NodeSize*, e.g. M (Medium size: 1xvCPU, 1 GB of RAM, 1 GB of ephemeral storage) or L (Large size: 2xvCPU, 2 GB of RAM, 1 GB of ephemeral storage). Nodes are further classified as *synchronous* or *asynchronous* depending on the behavior of the sub-application that the node virtualizes. Nodes are grouped into *NodeSets* - sets of nodes with identical functionality (i.e. nodes that virtualize the same sub-applications). Database server virtualization does not use containers and virtualizes structured and binary storage (uuObjectStore and uuBinaryStore) into a logical object called uuAppStore deployed to a virtual server.

4.2 uuGateway

Individual containers (nodes) are typically deployed on a public cloud infrastructure (e.g. Amazon AWS or Microsoft Azure) and access data from cloud-based data stores and databases (e.g. Mongo DB [23], etc.). Figure 1 illustrates the processing of client requests by the uuGateway. The uuGateway forwards the uuURI formatted client request to a router that passes the request to a load balancer. The load balancer selects a node from a NodeSet of functionally identical nodes, optimizing the use of the hardware infrastructure and providing a failover capability (i.e. if the node is not responsive the request is re-directed to an alternative node). uuURI is a version of a generic REST URI adapted for addressing uuApp applications. uuURI uses a standard string format to route the request to a specified gateway (e.g. Plus4U.net, etc.) and to specify which server should execute the request on behalf of the user. The URI header contains a special signed token that identifies an authorized user. The URI string has the following format:

<https://gateway/vendor-uuApp-uuSubApp-spp/tid-asid|awid/usecase>

where:

- gateway is the gateway address, e.g. Plus4U.net
- vendor code (e.g. Plus4U)
- uuApp – application (uuApp) code
- uuSubApp – sub-application (uuSubApp) code
- spp – optional spp (Separately Performing Parts) code within uuSubApp
- tid – tenant identifier
- asid – identifier of a specific sub-application instance
- awid workspace identifier
- usecase – use case identifier (i.e. API method)

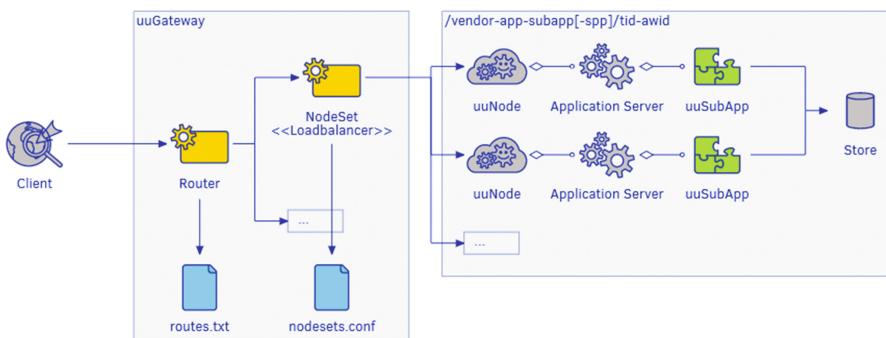


Fig. 1. uuGateway operation

In the case of static server resources, the use case identifier (usecase) is replaced by resourcePath (resourcePath) that points to a location where the resource is located.

4.3 uuCloud Operation Registry

Operation Registry is a key component of the uuCloud environment that maintains active information about all uuCloud objects (i.e. tenants, resource pools, regions, resource groups, hosts, nodes, etc.). uuCloud supports multi-tenant operation; each *tenant* typically represents a separate organization (e.g. a doctor's surgery). Tenants are assigned *resource pools* that define the maximum amount of hardware resources (i.e. number of vCPUs, RAM size and amount of storage) that are available for their operation. The Operation Registry records information about regions (e.g. Azure North (EU-N-AZ)), resource groups and hosts, and holds information about applications deployed to each node.

4.4 uuCloud Control Centre

The *Control Centre* includes tools for deploying and running applications in the uuCloud environment. Applications (nodes) are deployed into a *territory* (i.e. tenant) that is associated with a resource pool and managed using control centre tools. The control centre verifies permissions for the deployment of applications into a specific resource pool. Control centre tools are used to manage nodes, verify free resource pool capacity, locate a suitable host for the application, and to compile and deploy the node image on selected hosts.

5 Lekar Reservation System

To illustrate the application of the uuCloud framework we use the example of a recently implemented Doctor Surgery Reservation System - Lekar Reservation System (LRS). LRS (<https://www.plus4u.net/produkty-a-sluzby/lekar/>) is an on-line reservation system that manages communication between healthcare professionals (doctors, nurses, medical office staff) and patients in the Czech Republic. The main objective of the system is to enable registered patients to book a visit to any participating medical practitioner at any time (i.e. 24/7) using a mobile device or a computer, without having to phone the surgery during office hours to make an appointment. The LRS application generates automatic SMS and e-mail notifications to alert the patients of an upcoming appointment, processes reservation confirmations/cancellations, and generates reminders for regular check-ups. From the point of view of the health care professional, LRS provides an integrated diary showing appointments from all surgeries and gives a quick and easy access to basic patient information. The benefit for the patient is that all appointments are recorded in an easily accessible diary. Six months after the first release of the LRS application there were 210 active healthcare professionals with thousands of patients from across the Czech Republic using the system. As these numbers are expected to grow significantly in the future, the scalability of the system is a critical design consideration.

5.1 LRS Technical Solution

The LRS application covers three main business functions implemented as cloud services: Reservations, Organization & Management (i.e. registering and de-registering doctors, patients, nurses, and office staff) and Notifications (patient notifications via email and SMS messages). Additional *Common Services* that include user identification and authorization are provided by the UAF components running on uuCloud. The LRS system is implemented using open source technologies (Java, Ruby, MongoDB, Redis, Docker, Linux, etc.). The application is deployed to a hybrid cloud environment using the on-premise Plus4U.net platform (<https://www.plus4u.net>) in combination with MicroSoft Azure, demonstrating the technical feasibility and commercial benefits of implementing a large-scale system using open source and container-based technologies deployed on a hybrid cloud infrastructure.

5.2 Topology of the LRS Solution

Each doctor surgery or medical centre is assigned to its own *business territory* (implemented as a uuCloud tenant) with a separate ObjectStore (MongoDB database instance) ensuring data security by fully isolating individual cloud tenants. Table 1 shows the current allocation of NodeSets to the three main business functions. There are 19 nodes in total (1 Large and 18 Medium size nodes); all nodes are deployed on the Azure North MicroSoft data centre. Asynchronous nodes are used for email and SMS notifications, while the reservation sub-system uses synchronous nodes to perform booking of appointments. Figure 2 shows the topology of the LRS solution; the Application Deployment tenant is implemented on-premise in the MED-BT business territory of Unicorn Medical (Unicorn organization responsible for the development of healthcare information systems), while the runtime LRS system is deployed on the Microsoft Azure cloud platform using Docker virtualization technologies.

Table 1. Allocation of NodeSets to LRS business functions

Business Function	Reservations	Notifications	Organization & Management
NodeSet	1x Synchronous NodeSize L	11x Asynchronous NodeSize M	2x Synchronous 5x Asynchronous NodeSize M

The present version of the LRS system is the first step towards a more comprehensive outpatient system for individual medical practitioners and large healthcare facilities. The inherent scalability of this cloud solution makes it possible to add computing and storage resources as the need arises with minimal incremental cost while maintaining high availability and response time characteristics.

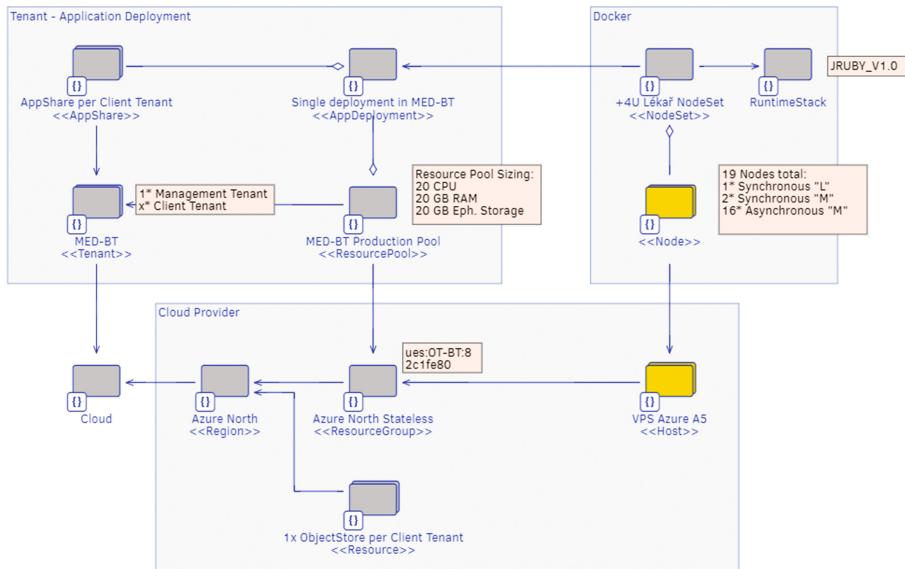


Fig. 2. Topology of the LRS cloud solution

6 Conclusions and Further Work

We have argued that the accelerating shift towards cloud computing combined with the dominance of mobile computing and the growing use of IoT devices requires a re-assessment of the existing architectural principles and the associated application development frameworks. While the use of micro-services and container-based virtualization brings many benefits, the highly distributed nature of the resulting applications and the short software release cycles present many challenges to organizations involved in the development of cloud-based enterprise applications. A suitable application development framework and associated methods and tools are an essential pre-requisite for achieving successful project results on a repeatable basis. The uuCloud platform described in this paper was developed specifically to address the requirements of cloud-based applications and is used currently for the development of large-scale enterprise applications at Unicorn. We have illustrated the application of uuCloud using a *real-world* Doctor Surgery Reservation System that is used by hundreds of healthcare professional and thousands of patients across the Czech Republic.

There are some similarities between uuCloud and other frameworks such as Cloud Foundry, OpenShift and Kubernetes. Kubernetes, in particular supports similar functionality as uuCloud. We have evaluated these frameworks before starting the UAF project and decided that our needs would be best served with a fully integrated architecture that incorporates uuCloud, uuUserInterface and uuIoT frameworks (see UAF description in Sect. 3). The uuCloud framework currently supports Docker containers and Microsoft Azure cloud infrastructure, but we are extending the framework to incorporate other technology solutions. Our present efforts focus on improving

the monitoring tools and ensuring high availability and good response time for applications running on public cloud infrastructure. We are investigating the feasibility and cost of deploying applications across multiple availability zones, with the aim of providing users with similar SLA (Service Level Agreement) guarantees as we are able to provide for on-premise applications. We are continuously monitoring the rapidly evolving landscape of cloud platforms and frameworks. We may decide to align the uuCloud framework with Kubernetes in the future as both frameworks mature and the direction of cloud standardization becomes clearer.

References

1. Amazon.com (2017). <http://aws.amazon.com/>. Accessed 7 July 2017
2. Microsoft Azure: Cloud Computing Platform & Services (2017). <https://azure.microsoft.com/en-au/>. Accessed 22 Aug 2017
3. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Microservices architecture enables DevOps: migration to a cloud-native architecture. *IEEE Softw.* **33**(3), 42–52 (2016)
4. Beránek, M., Feuerlicht, G., Kovář, V.: Developing enterprise applications for cloud: the unicorn application framework. In: International Conference on Grid, Cloud and Cluster Computing, GCC 2017, Las Vegas, USA. CSREA Press (2017)
5. Thönes, J.: Microservices. *IEEE Softw.* **32**(1), 116 (2015)
6. Mahmood, Z., Saeed, S.: Software Engineering Frameworks for the Cloud Computing Paradigm. Springer, London (2013). <https://doi.org/10.1007/978-1-4471-5031-2>
7. Namiot, D., Sneps-Sneppe, M.: On micro-services architecture. *Int. J. Open Inf. Technol.* **2** (9), 24–27 (2014)
8. Splitting Applications or Services for Scale, AKF Partners. <http://akfpartners.com/growth-blog/splitting-applications-or-services-for-scale>. Accessed 22 Aug 2017
9. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Migrating to cloud-native architectures using microservices: an experience report. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 201–215. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33313-7_15
10. Rimal, B.P., et al.: Architectural requirements for cloud computing systems: an enterprise cloud approach. *J. Grid Comput.* **9**(1), 3–26 (2011)
11. Raj, P., Venkatesh, V., Amirtharajan, R.: Envisioning the cloud-induced transformations in the software engineering discipline. In: Mahmood, Z., Saeed, S. (eds.) Software Engineering Frameworks for the Cloud Computing Paradigm, pp. 25–53. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5031-2_2
12. Ramachandran, M.: Business requirements engineering for developing cloud computing services. In: Mahmood, Z., Saeed, S. (eds.) Software Engineering Frameworks for the Cloud Computing Paradigm, pp. 123–143. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5031-2_6
13. Feuerlicht, G., Thai Tran, H.: Adapting service development life-cycle for cloud. In: Proceedings of the 17th International Conference on Enterprise Information Systems, vol. 3. SCITEPRESS-Science and Technology Publications, Lda (2015)
14. Tran, H.T., Feuerlicht, G.: Service repository for cloud service consumer life cycle management. In: Dustdar, S., Leymann, F., Villari, M. (eds.) ESOCC 2015. LNCS, vol. 9306, pp. 171–180. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24072-5_12

15. CloudFoundry. Cloud Application Platform - DevOps Platform, Cloud Foundry (2017). <https://www.cloudfoundry.org/>. Accessed 28 Sep 2017
16. OpenShift: Container Application Platform by Red Hat, Built on Docker and Kubernetes (2017). <https://www.openshift.com/>. Accessed 28 Sep 2017
17. Kubernetes (2017): <https://kubernetes.io/>. Accessed 25 Aug 2017
18. Burns, B., et al.: Borg, Omega, and Kubernetes. Queue **14**(1), 70–93 (2016)
19. Home - Cloud Native Computing Foundation (2017). <https://www.cncf.io/>. Accessed 27 Sep 2017
20. Truyen, E., et al.: Towards a container-based architecture for multi-tenant SaaS applications. In: Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware, pp. 1–6. ACM, Trento, Italy (2016)
21. WC3. HTML5. <https://www.w3.org/TR/html5/>. Accessed 21 Aug 2017
22. Docker. What is Docker. 2015 2015-05-14. <https://www.docker.com/what-docker>. Accessed 21 Aug 2017
23. MongoDB for GIANT Ideas (2017). <https://www.mongodb.com/index>. Accessed 21 Aug 2017



A Knowledge Carrying Service-Component Architecture for Smart Cyber Physical Systems

An Example Based on Self-documenting Production Systems

Christopher Haubeck¹(✉), Winfried Lamersdorf¹, and Alexander Fay²

¹ University of Hamburg, Hamburg, Germany

{haubeck, lamersdorf}@informatik.uni-hamburg.de

² Helmut Schmidt University, Hamburg, Germany

alexander.fay@hsu-hh.de

Abstract. Cyber Physical Systems (CPSs) are both software and hardware intense systems which are integrated into the digital world. An increasingly relevant application area for CPSs are software-driven industrial Production Automation Systems. A major driver of these Cyber Physical Production Systems is the need to integrate intelligent and smart functionalities into the production process at runtime that can run in the cyber world and do not radically change the underlying control flow of the physical plant. To gather the necessary knowledge for smart functionalities both cyber and physical parts must be captured, kept and analyzed together in a suitable CPS architecture – also despite frequent evolutionary changes during their whole (usually relative long) life span. This paper addresses this problem by presenting a service-component architecture for smart service-oriented CPSs which can manage implicit knowledge by acquiring, handling and analyzing domain specific service-components with respect to the CPS context.

Keywords: Cyber Physical Systems · Service-component architecture
Self-awareness · Models@runtime · Production Automation
Distributed systems

1 Introduction

Cyber-Physical Systems (CPSs) represent a new paradigm to describe the close interrelationship between hardware and software components. One motivation for this interconnection is to bring intelligence and smarter functionalities into traditional, hardware-dominated application areas such as embedded systems, robotics, or system networks [1]. As an example, this contribution considers the practically relevant area of Production Automation which can be seen as early CPSs [2]. The most important part to reach the full dimension of CPSs is to integrate communication and information gathering capabilities that allow pursuing goals and functionalities on a global system level [3]. In contrast to that, in traditional production systems information is often not easily accessible. Because here the overall knowledge is only implicitly available in the

respective control code as well as the hardware setup. This is a result of the industrial practice to often use only informal specifications for implementing system control [4]. Plants are long-living and constantly evolving over their live span. Therefore, documentation is nearly never up-to-date, which complicates (or even prevents) the task of analyzing the production system [18]. Since, however, each production system is based on the respective companies' application requirements, it is desirable to make the system analyzable regarding its requirements only by itself. Here, services allow gluing the physical production process together with software-driven capturing functionalities. Further, functionalities captured in services allow forming a self-documenting process as proposed for the need of evolution support in this contribution. Correspondingly, the aim of this contribution is to tackle the lack of documentation with service-oriented methods to integrate production systems into the future world of smart CPSs.

The paper is structured along three questions: (Q1) How can the behavior of joint dynamics between software and hardware be captured in evolvable service-components? (Q2) Can these service-components provide a basis for reducing the lack of documentation? (Q3) How can a comprehensive process for self-documenting be implemented by assembling reusable services? To address these questions, a service-component architecture is presented which is based on event-based monitoring in autonomous service-components. First, the problem of a consistent system state in service-components of a CPS in general and a production system in particular is addressed. Following that, the paper shows how domain-specific models are captured in and managed by service-components to represent the combined hardware and software behavior. Finally, the paper shows how these service-components can be used to implement a self-documenting process. The approach is evaluated by two case studies which demonstrate the proposed concept. Related work and a conclusion close the paper.

2 Service-Oriented Knowledge Management in CPS

The understanding of a CPS is not clearly defined and various kinds of definitions from different domains exist. In this contribution, the architecture for a smart service-oriented CPS is presented based on Sztipanovits et al. [5] who define CPS in three different layers: As shown in Fig. 1, the lowest level is the physical connection layer which consists of physical components and their interactions. In production systems, these are mechanical parts which are pneumatically (thick lines in Fig. 1) or electrically (dashed lines) connected. The other two layers are the cyber layers: first the cyber connection layer which consists of software components embedded in the hardware system. In terms of production systems these are the execution environments and their software code of Programmable Logic Controllers (PLC) which are connected by bus-systems. To control the plant, events of actuators transfer the code behavior into physical behavior while events of sensors transfer the physical results back. The last layer is implemented in ordinary software languages (e.g. object-oriented languages) and capable of CPS functionalities (e.g. network communication). In this paper this layer focuses on smart functionalities of CPSs and especially self-documentation. It is further named the *knowledge carrying software* layer.

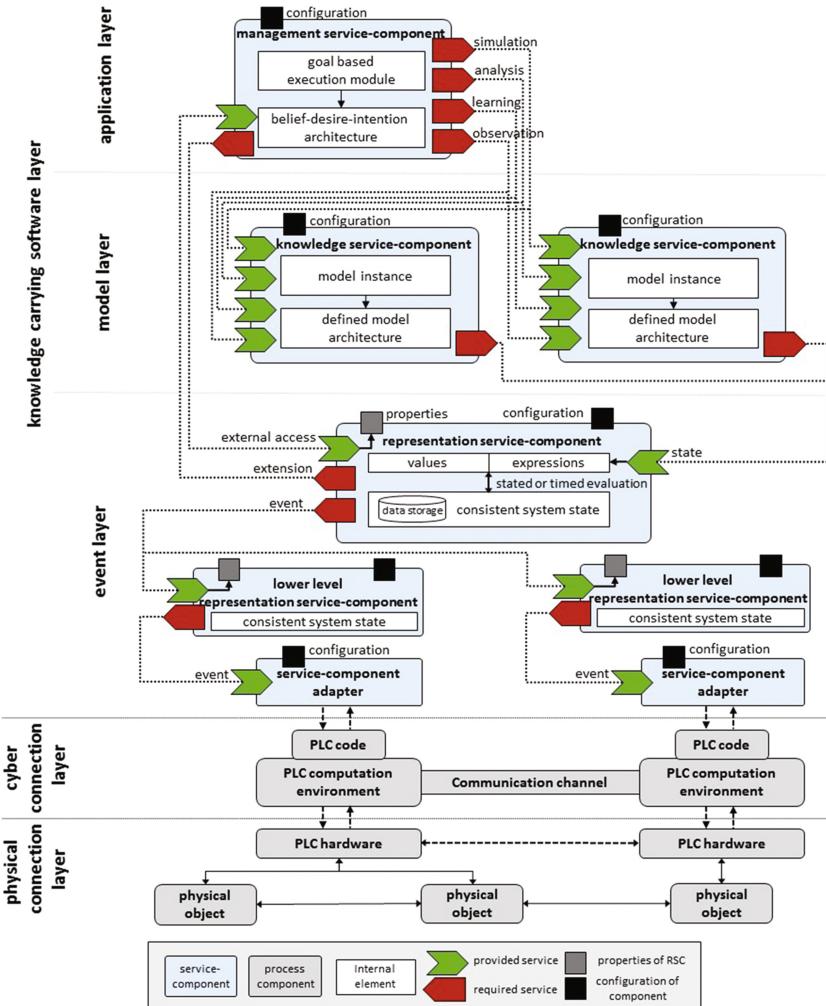


Fig. 1. Service-component architecture for smart CPS

For self-documenting, the CPS architecture follows the general guideline from Robinson for a requirement monitoring system [6] by applying it to CPSs and the service-oriented world. Accordingly, the knowledge carrying software layer consists of the event, model and application layer as shown in Fig. 1. Within the event layer events are acquired, (pre-) processed, and stored in a state service (Sect. 2.1). Due to the interdisciplinary interactions of the underlying process in case of a CPS, the model layer holds domain specific models in knowledge service-components to determine process properties (Sect. 2.2). The application layer contains application specific components and is in this contribution discussed with respect to the targeted application of a self-documenting process for evolution support (Sect. 2.3). The visualization layer

of Robinson is not discussed here. In a nutshell, this contribution proposes a service-oriented component architecture that captures behavior in models on top of traditional hardware systems so that they become smart within a service-oriented CPS.

2.1 Service-Oriented Event Acquisition in Heterogeneous CPSs

To follow compositional design architectures, elements should communicate homogeneously regarding their composed properties and semantics [5]. But unfortunately, CPSs differ in structure and behavior rather heterogeneously [5]. In addition, production systems are generally written in domain specific languages and always run cyclic in their own program thread and under hard real-time conditions. This implies that only event sequences are shown as an interface towards the upper CPS cyber levels and the actual execution remains as a black box. These facts and the general conservatism of the production industry make it problematic or even impossible to couple the component of the production process directly with the CPS by weaving code into the controlling software like most other monitoring approaches propose [6]. From the authors' point of view, this difficulty of external access holds for many other domains of CPS as well. Therefore, this paper proposes a service-oriented approach which captures information of event sources externally in a non-invasive way.

Since smart functionalities often require a high level of autonomy, the proposed architecture is based on *active service-components* (see [7]). In a CPS view, active components are software representations of (hardware) entities which are managed and defined like hierarchical components that communicate with each other via service calls and provide a flexible internal architecture to act autonomously [7]. For event acquisition, the architecture distinguishes between two types of components:

First *service-component adapters (SCA)* are deployed in a 1-to-1 mapping per available entry points of the CPS (see Fig. 1). They allow event sources to participate in the CPS by providing services within the event-acquisition. These services behave as proxies that allow for a transparent access to the connection layers and encapsulate the generation and propagation of events. To allow a high degree of freedom, they operate autonomously by actively establishing and managing connections to the connection layer with domain specific technologies like process control protocols. Further also pre- and post-processing steps like caching are done. In CPSs it is important to decouple the different layers – although it might affect the performance. Consequently, the utilization of provided service- components reduces the complexity in terms of crosscutting interactions between the layers. Services allow cyber components to browse and register for events out of the event layer. Further, the architecture decouples the rate of the events provided into the CPS (desired rate) with the rate coming from the connection layer (source rate) inside the service. Note that this decoupling implies the assumption that values do not change between two measured events according to the source rate.

The second type of components in the event-acquisition layer are *representation service-components (RSC)*. As shown in Fig. 1 the separately deployed RSCs are in charge of the consistent system state by requiring the services of the SCA. All RSCs in their hierarchical order are further called the *system façade*. RSCs can run in different computing nodes as SCAs, because in contrast to SCAs, the upper cyber layers of CPSs should follow a business-oriented modularization. Such a modularization can be made

per technical resources, but can differ by following, e.g., a business process. Nevertheless, for a consistent system façade each RSC must be responsible for a separated part of the system to have a clear responsibility chain on the cyber layers.

2.2 Domain Specific Models for Joint Dynamics in Service-Components

In complex systems – especially with hardware relations – models play an increasing role for different phases of the lifecycle as Derler et al. stated also for CPSs [1]. Therefore, the employment of models is also necessary for service-oriented CPSs that implement smart functionalities at runtime [2]. For CPSs, generally models of the behavior are considered [1]. The heterogeneity and complexity of these models stresses modeling languages of CPSs [1] and therefore it is problematical to find a uniform architecture to integrate such models. To tackle this difficulty, this paper proposes to encapsulate each model in its own design architecture and execution environment that is only accessible by services. These so-called *knowledge service-components* (KSC) can exist on each hierarchical level of the system façade as shown in Fig. 1. KSCs describe the CPS behavior and determine high-level properties. The event-acquisition in RSCs and the models in KSCs must be operated by services together. For such an event-based service system this contribution establishes an approach on policy-based evaluated statements for service calls. These statements consist of three parts:

1. Each statement has a *state expression* which is based on the state service of the RSC. An expression of a service call may consume several events, and the RSC computes the expression's value. Expressions are evaluated within the RSC to guarantee that evaluation takes place in the safeguarded system facade.
2. Statements use *matchers* for expressions that check the expression value against a desired result. An expression value is only returned to the service caller when the matcher returns a positive result.
3. Since the evaluation is separated in the RSC and the time when the statement is fulfilled is unknown, statements have an *evaluation policy*. Policies determine an interval for the evaluation of expressions. This means the RSC asks the policies when the next evaluation should take place by scheduling a task to evaluate and match the expression when a new and consistent state of involved events is available. Intervals can be linear, accelerating or decelerating towards a deadline.

In conclusion, models in KSCs define a pool of statements described by event expressions and transfer them with a policy and matcher to a RSC. An approach that can also be used as an alternative to the often-used pull principle in other event-based domains. For example, test cases of events which occurrence time is unknown can be better defined by this approach. Returned values of statements are processed by so-called *mode services* that determine how statements can affect a KSC (see Fig. 1). This contribution proposes a not necessarily complete list of four mode services:

1. *Learning services* provide learning mechanisms within the KSC execution environment which build up the model. Although simpler models like algebraic functions may not require learning, complex models must use learning algorithms.

2. An *observation service* describes how a model acts during live observation. This is necessary whenever the KSC has statements of the current system state and wants to reflect this state in its model. Further *anomalies* which are differences between the model and the actual observed state seen in the system can be detected here.
3. An *analysis service* describes how high level properties necessary, for example, in the self-documenting process are determined based on the model structure and state. Here the advantage of model-based service-components for CPSs shines since formal properties can be expressed as services that can be determined automatically.
4. The last *simulation service* is the most optional one and can be used in addition to the analysis. Simulation allows the KSC to predict the behavior of models in terms of already observed situations. Therefore, often observed usage scenarios are extracted to replay these scenarios to evaluate the model in typical situations.

2.3 Self-documenting Process for Evolution Support

As an application example, this section shows a self-documenting process for evolution w.r.t non-functional properties. The process is shown in Fig. 2 and allows tackling the problem of constant evolution in long-living systems which often results in non-documented behavior and unknown system properties [18]. To support evolution, event traces of the system are monitored. As a side note, also events from test cases can be used here (see [17]). To provide analysis, events must be enriched with a machine interpretable *semantic context*. For example, for production systems, each event holds an assigned event-context identifier for the topology and the type of events in a tree structure. The topology part describes from which system part an event stems. The type part tells which information it gives. The behavior is documented in the RSC hierarchy that holds KSCs which consume timed events in mode services as shown previously. When the observation service detects changes, a human in the loop decides if the behavior is desired or undesired. Desired changes are considered as an evolution and undesired as an anomaly. The models are analyzed to find high-level information of the technical process, e.g. invariants of events or (non-functional) properties. Further, simulation of usage scenarios can be used for analysis. In general, the process actively reduces degeneration of documentation by coming from an undocumented system to a system which is documented by a model-based specification.

The self-documenting process is demonstrated in this paper, but the service-component architecture can be extended to other functionalities. These functionalities are situated in the application layer. To implement the process RSCs are extended with a *management service-component* (MSC) (see Fig. 1) that also communicates via services. When considering CPSs with high dynamics, a robust goal-based approach seems most favorable. This preference results from the fact that the process is, in general, based on priorities of the mode services whose priorities depend on the current condition of the underlying system. For example, the observation mode is continued until a change occurs and learning is necessary. One suitable architecture is a “goal-based approach with deliberation according to priorities” following a Belief-Desire-Intention architecture [8]. The main goal hierarchy (omitting plans, beliefs, and smaller goals) is shown in Fig. 3 (left). The overall goal to ensure documentation is enhanced by sub-goals for mode services of the KCSs. When a

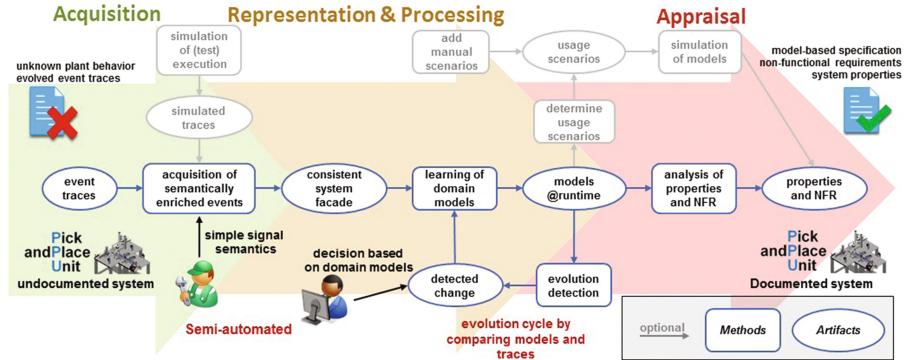


Fig. 2. Evolution support process to document behavior of production systems

specific condition is fulfilled, a goal deliberation takes place and other mode services can suppress the current service. In this way, the most needed service is always activated. Human interaction is included for decisions. This semi-automated approach is chosen because of domain restrictions, but could also be substituted by adding additional service-components.

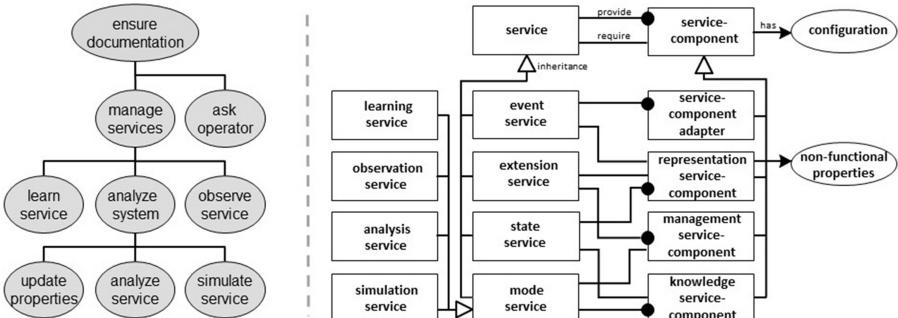


Fig. 3. (left) Goal hierarchy of the process; (right) components and services view

3 Evaluation

To evaluate the concept of a smart service-oriented CPS two case studies are performed: one demonstrates the ability to handle hierarchical event sources and the second applies the incremental documentation process for evolution support. The respective prototype was implemented in Java based on active components of the Jadex platform [7]. Main components and services of the prototype are shown in Fig. 3 (right). The plant connection is done with the industrial M2M standard OPC implemented in a SCA that communicates with RSCs via event services implemented within a stream-oriented programming style based on the Jadex “Future” concept (see [7]).

To ensure consistency in the system façade, RSCs follow the BASE principle (for: basically available, soft state, eventual consistency) by using the timestamps of the connection layers. Each component holds a configuration and RSCs hold the analyzed properties of their corresponding plant part. MSCs extend the RSCs via extension services by managing the self-documenting process with an implementation based on the BDI-Kernel of Jadex (see [8]). KSCs have one of two domain models developed to ensure a high degree of expressiveness w.r.t documentation (see [18]). The models are implemented as petri-nets in the Jadex micro-kernel and communicate via state service to the RSCs and via mode services implemented for both models to the MSCs.

A. Case Study: Hierarchical Distributed Production System

For evaluation of hierarchical CPSs and distributed event sources, a laboratory plant on the Helmut-Schmidt University was used. It consists of three distinct producing machines, an arm robot, a high rack storage area and a portal crane. An intra-logistics transport system of conveyor belts divided into five subsystems connects the elements. For self-documenting purposes, models for machine states are used in the learning services (see [18]) to create models expressing the state of subsystems. Each RSC creates its own separated state model. Regarding documentation, the case study considers the nominal capacity of the logistic system for each conveyor, subsystem and whole plant as a non-functional property. The analysis service uses a formal algorithm specified on the domain state model and combines the capacity of the whole plant on the parent RSC by an aggregation formula. The results are shown in Table 1. The service-oriented CPS shows that it can ensure a consistent hierarchical system view, because the properties and a detailed formal model documentation are constantly reflected in services. Such properties bound to documenting service-components are useful by themselves, because otherwise they are, if at all, only manually determined.

Table 1. Results of the experiments regarding capacity

Resource	Mean time [s]	Standard deviation [s]
Conveyor 1.0 (SS1)	4.30	0.02
Conveyor 1.1 (SS1)	4.22	0.04
Conveyor 1.2 (SS1)	3.89	0.05
Conveyor 1.3 (SS1)	12.78	0.08
Subsystem 1, aggregated	25.19	0.1
Subsystem 2, aggregated	26.78	0.09
Subsystem 3, aggregated	36.33	0.72
Subsystem 4, aggregated	47.36	0.74
Subsystem 5, aggregated	64.07	21.85
Whole plant, aggregated	199,73	21.87

B. Case Study: Evolution of a manufacturing system

A second case study was conducted on a Pick-and-Place-Unit (PPU) at the Technical University in Munich (see [9]). The PPU serves as a demonstrator for automated

production plants under evolution. It is capable of processing different workpiece types, namely metallic and plastic workpieces. Metallic workpieces are to be processed by stamping them. The PPU consists of four parts: (1) A magazine that provides workpieces to be processed. (2) A stamp that stamps metallic workpieces. (3) An output ramp for releasing workpieces from the plant. (4) A crane which connects the other plant parts. Evolution scenarios built up the PPU consecutively (names according to [9]). In scenario Sc1, the PPU just consists of the magazine and the crane which is extended by workpiece identification in Sc2. In Sc3, a stamp is added for metallic workpieces and the last considered Sc5 tries to optimize the throughput by a modified crane behavior. In comparison to the previous case study which captured the machine state of the system, here the actual routing of products is captured by filtering events according to the assigned context semantics (see [18]). This case study demonstrates that the self-documenting process based on the RSCs can be fully implemented. The resulting petri net is shown in Fig. 4. Every scenario is presented in different line types. Timing information is shown whenever it changes (just in Sc5). Identifying events are shown in braces. The self-documenting process detects changes in the observation service and then relearns the model per new scenario. The model shows how the KSC prepares the documentation for an engineer and that he is able to analyses the system evolution with formal petri-nets which are often used in the industrial context as documentation.

Every time the KSC changes, the process demands an analysis on the model structure. Table 2 shows the result of some typical properties. Here, the benefit of constant relearning in service-components can be seen, because improvements and influences of evolution can be directly inspected. For example, it can be easily seen that Scenario 5 does not improve the mean transportation time as stated in [9] and only reduces the standard deviation. Further on, also the reason for this lack of performance which is a low horizontal crane speed is reflected in the routing model of the KSC. Overall the case studies verify the answers to the given questions of this paper. Because they show that evolvable service-components can encapsulate knowledge of CPSs in models (see Q1) and serve as a documentation for the engineer (see Q2). And the composed self-documenting process of this services (Q3) allows the service-oriented CPS to identify that the evolution change have failed and should be reversed.

4 Related Work

This contribution provides an approach which combines different methods. One way for processing event-traces is complex-event-processing (CEP). As one example, Michelsen proposes data stream processing in decentralized networks [10], but as other CEP approaches do not consider acquisition based on complex models. This is also the case for approaches which use learning techniques to find dependencies between events. Margara, for example, generates rules for event dependencies by using historical traces [12]. However, general approaches for (distributed) event-based system normally rely on query languages and mathematical aggregation and result in event rules. They do not consider the wide range of models needed for CPSs, nor do they consider models in service-components. Runtime monitoring is a commonly used

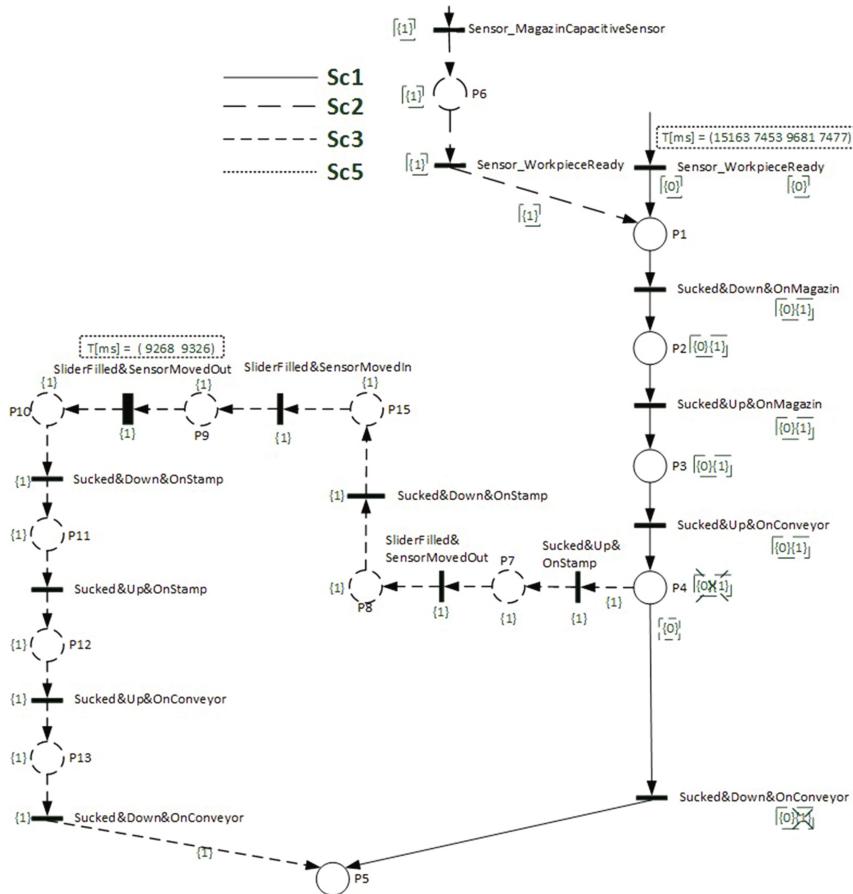


Fig. 4. Routing petri-net of scenario 1, 2, 3 and 5 of the PPU case study

Table 2. Results of the incremental analysis

Indicator	Sc1	Sc2	Sc3	Sc5
Mean transport time	7816 ms	7807 ms	2102 ms	3802 ms
Standard deviation of transport time	1373 ms	1736 ms	4045 ms	1600 ms
Number of routes	1	1	2	2
Number of places	5	6	14	14
Number of transitions	5	7	16	16

method. An overview is given in [13] and for CPSs it is shown in [14]. Most monitoring approaches of CPSs try to tackle the problem of how provided specifications can be used (see [14]). The approach presented here aims at models which can be learned within a system by service-component. Further, related work also exists for the different mode services of the models. Although these algorithms usually rely on

predefined models, our approach is based on similar models and algorithms developed by the authors in these areas. Nonetheless, other algorithms can be integrated as additional KSCs.

Service-oriented CPSs are also mentioned in the literature. But as stated in the survey of Hoang et al. [11] service-oriented CPSs are “typically designed to coordinate the computational and physical parts of a system”. These approaches often use middleware approaches for CPS. For instance, in [19] a semantic middleware is presented that integrates various CPS elements. This often-focused topic of heterogeneity is also present in the here presented approach, but this paper rather enables high-level model analysis and knowledge gathering in component-services. To categorize the approach in the taxonomy of Hoang et al. [11] this contribution follows a distributed architectural topology with a strong focus on the application layer to provide high-level management.

Production systems are considered in service-oriented architectures mainly for accessing control of manufacturing resources. For example, Valilai et al. aim at resource sharing by supporting collaboration and data integration [16]. Also, real-time services are considered. Lin et al. [20], for instance, proposes these for sustainability and predictability of CPSs. Our approach does not consider resource sharing or real-time services with services, but rather capture behavior. Similarity exists here to algorithms of fault detection that create models like discrete event models or process models [15] in order to describe the monitored system. But these approaches do not consider software layers provided by CPSs and approaches are limited to the production automation domain. Four as-a-service (*aaS) models are identified in cloud manufacturing. Cloud is not directly considered in this paper, but on the cyber levels services are comparable to cloud services following mostly a software-as-a-service model. Nevertheless, our research questions regarding services do not aim at typical cloud systems.

In summary, parts of the approach are considered in different areas and the approach partly relies on these methods. Also, service-oriented CPSs are present in the literature. But existing approaches do not consider service-oriented architectures for the automatic gathering of knowledge, nor are these approaches related to the needs of a service-based learning integrating all layers of a CPS. To our best knowledge, architectures for combining distributed event-monitoring, model-based learning and analyzing of CPSs in service-components are not present in the current literature.

5 Conclusion

This paper presented Production Automation as an important application domain in which service-oriented CPSs are worth to be analyzed, tested and implemented. The paper showed specific problems of CPSs, like capturing joined dynamics, building domain-specific models, and implementing smart functionalities. To address these problems, the paper proposed a service-oriented component architecture. The layered CPS architecture can integrate diverse types of event sources by using autonomous service-component adapters while maintaining a service-component façade in a CPS environment. The contribution showed how corresponding knowledge service-components can be managed in the façade and how services modifying the integrated

models can be combined to smart CPS functionalities. This was shown by presenting a self-documenting process based on mode services that supports evolution of the underlying system. The approach was evaluated on a distributed as well as evolving production plant.

Future work can consider non-event-based CPSs and service-based synchronization of higher abstracted KSCs. Further, an extended approach can be imagined that exploits the generated documentation by exchanging documentation between different service-based CPSs in order to actively provide recommendations.

Acknowledgment. This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future.

References

1. Derler, P., Lee, E.A., Vincentelli, A.S.: Modeling cyber–physical systems. Proc. of the IEEE **100**(1), 13–28 (2012). Modeling cyber–physical systems
2. Lee, E.A.: Cyber Physical Systems: Design Challenges. Berkeley (2008)
3. Dumitache, I., Caramihai, S.I.: Intelligent cyber-enterprise in the production context. In: World Congress of the 19th International Federation of Automatic Control (2014)
4. Frey, G., Litz, L.: Formal methods in PLC programming. In: IEEE International Conference on: Systems, Man, and Cybernetics (2000)
5. Sztipanovits, J., Koutsoukos, X., Karsai, G., Kottenstette, N., et al.: Toward a science of cyber-physical system integration. Proc. of the IEEE **100**(1), 29–44 (2012)
6. Robinson, W.N.: A requirements monitoring framework for enterprise systems. Requir. Eng. **11**(1), 17–41 (2006)
7. Braubach, L., Pokahr, A.: Developing distributes systems with active components and Jadex. Sci. Int. J. Parallel Distrib. Comput. **13**(2), 100–119 (2012)
8. Pokahr, A., Braubach, L., Haubeck, C., Ladiges, J.: Programming BDI agents with pure Java. In: Müller, J.P., Weyrich, M. (eds.) MATES 2014. LNCS (LNAI), vol. 8732, pp. 216–233. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11584-9_15
9. Vogel-Heuser, B., Legat, C., Folmer, J., Feldmann, S.: Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the Pick and Place Unit (2014)
10. Michelsen, T.: Data stream processing in dynamic and decentralized peer-to-peer networks. In: SIGMOD Ph.D. Symposium, pp. 1–5 (2014)
11. Hoang, D., Paik, H.Y., Kim, C.K.: Service-oriented middleware architectures for cyber-physical systems. Int. J. Comput. Sci. Netw. Secur. **12**(1), 79–87 (2012)
12. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the past: automated rule generation for complex event processing. In: International Conference on Distributed Event-Based Systemspp, pp. 47–58 (2014)
13. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection for discrete sequences: a survey. Trans. Knowl. Data Eng. **24**(5), 823–839 (2012)
14. Mao, J., Chen, L. (eds.): Runtime monitoring for cyber-physical systems: a case study of cooperative adaptive cruise control. In: Intelligent System Design and Engineering Application (2012)
15. Schneider, S., Litz, L., Danancher, M.: Timed residuals for fault detection and isolation in discrete event systems. In: Dependable Control of Discrete Systems, pp. 35–40 (2011)

16. Valilai, O., Houshmand, M.: A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm. *Rob. Comput. Integrat. Manuf.* **29**(1), 110–127 (2013)
17. Ladiges, J., Fay, A., Haubeck, C., Lamersdorf, W., Lity, S., Schaefer, I.: Supporting commissioning of production plants by model-based testing and model learning. In: International Symposium on Industrial Electronics, pp. 606–611 (2015)
18. Ladiges, J., Haubeck, C., Fay, A., Lamersdorf, W.: Evolution management of production facilities by semi-automated requirement verification. *at-Automatisierungstechnik* **62**(11), 781–793 (2014)
19. Franke, M., Seidl, C., Schlegel, T.: A seamless integration, semantic middleware for cyber-physical systems. *Network, Sensing and Control*, pp. 627–632 (2013)
20. Lin, K.Y., Panahi, M.: A real-time service-oriented framework to support sustainable cyber-physical systems. In: International Conference on Industrial Informatic (2010)



Experiences on Migrating RESTful Web Services to GraphQL

Maximilian Vogel^{1,2(✉)}, Sebastian Weber², and Christian Zirpins¹

¹ Karlsruhe University of Applied Sciences, Moltkestr. 30, 76133 Karlsruhe, Germany
`christian.zirpins@hs-karlsruhe.de`

² diva-e Nettioneer GmbH, Ludwig-Erhard-Allee 20, 76131 Karlsruhe, Germany
`{maximilian.vogel, sebastian.weber}@diva-e.com`

Abstract. Web service APIs are central hubs of modern cloud-based application systems. Over recent years, REST has become a de facto standard for their architectural style. Yet in scenarios like mobile apps, flexible client-centric data fetching approaches have emerged as a promising alternative. This gives rise to the question whether RESTful systems can be migrated to a technique like GraphQL and benefit from the new approach. In this paper we report on our experiences during such migration of a real world smart home application. Our observations have underpinned some of the conceptual benefits but also identified challenging aspects where further research is required.

Keywords: Software service architecture · REST · GraphQL

1 Introduction

Many modern service-oriented systems – especially those based on microservice abstractions [14] – embrace the REST-style of resource-oriented distributed software architecture [8]. Beyond offering a simple and solid solution for developers to realize interprocess communication, the REST architectural style leads to some favorable characteristics for distributed systems. Among them are *loose coupling* and *composability* of services as well as *flexibility*, *robustness* and *scalability* of resulting systems.

REST promotes the decomposition of systems into sets of linked resources with a certain level of granularity. This leads to difficult trade-offs between reusability and performance that are well known in general software service architecture (e.g. [4, 7]). We generally seek less granular and more cohesive (micro) services fostering loose coupling and high reusability. But this might require cumbersome client server conversations with multiple consecutive requests traversing the resource graph (aka “under-fetching”). The inverse approach of the *Coarse-Grained Remote Interfaces* pattern trades fewer requests with less networking overhead against lower cohesion and reusability [13]. Moreover, this often leads to the transfer of too much data for individual requests (aka “over-fetching”).

Finally, service granularity might be optimized for specific use cases, but these are hard to reuse triggering an ongoing proliferation of variants.

Beyond architectural quality, another point to consider is the *ease of using service APIs* from a client perspective. As Byron puts it, app developers “*... don't think of data in terms of resource URLs ... (but) in terms of a graph of objects ...*” relevant to their apps [1]. This has led to a number of *declarative data fetching languages* like Facebook's *GraphQL* [5]. GraphQL enables app developers to run queries against a server API that traverse a graph of application objects and fetch exactly the right amount and format of data in a single request. Such APIs omit over- as well as under-fetching without the need for specialized services, thus yielding performance benefits while being widely reusable.

The potentials of declarative data fetching have recently triggered discussions on whether a technology like GraphQL is an alternative (and thus in contradiction) or a possible extension (and therefore in conformance) to the REST architectural style and its implementing frameworks. This discussion is also relevant for the case of existing RESTful systems with respect to possibilities of their modification in order to benefit from declarative data fetching. To this end, questions arise on two different levels:

- Individual Software Services
 - How can existing RESTful services be transformed or extended to offer declarative data fetching capabilities?
 - What are the performance costs and technical challenges of query execution and are they always justified?
- Service-Oriented Systems and Architecture
 - Are services and systems with declarative data fetching capabilities (still) RESTful and exhibit the related architectural benefits?
 - Which architectural design patterns emerge for such systems?

In this paper we don't seek to finally answer the above questions. Instead, we offer an experience report along with a discussion on the migration of purely REST-based software services to GraphQL. We show an industrial case study that builds on a smart home platform and demonstrate the consequences of migrating parts of its RESTful API to GraphQL. Our results underpin the conceptual advantages of declarative data fetching in practice. Yet, we have identified challenges on different levels that should be considered during migration.

The rest of this paper is structured as follows: In Sect. 2 we first discuss related work. We then give a short introduction to GraphQL in Sect. 3. Based on this background, we present our case study on migrating REST-based systems to GraphQL in Sect. 4. Next, we discuss challenges of migration on the level of service-oriented systems and architecture (Sect. 5.1) as well as individual software services (Sect. 5.2). Finally in Sect. 6, we summarize our work and give an outlook.

2 Related Work

From an academic standpoint, little research has been published specifically about GraphQL yet. Currently, Facebook provides only an informal documentation of GraphQL [5]. Beyond that, Hartig and Pérez have studied the language from a theoretical perspective and provided a formal query semantics [10].

Falcor by Netflix and *OData (Open Data Protocol)* are flexible data fetching libraries and constitute GraphQL alternatives [11, 17, 18]. Helfer categorizes them as Web API technologies and brings them into a chronological sequence with REST and SOAP [11]. Cupek and Huczala compared OData with REST [3] but not with GraphQL. Falcor as well as GraphQL are JSON-oriented technologies that both emerged in 2015 and act as middleware to query backend data sources [15, 16]. At the time of writing, no academic work exists on Falcor. However, several web-based resources provide comparisons of Falcor, OData and GraphQL [12, 15, 16]. As Meredith concludes, “*The Falcor data model is a graph, and the GraphQL data model is a tree.*” [15]. In this paper, we solely focus on GraphQL.

Recently, some student research has emerged on the topic. The thesis by Cederlund is in the field of performance and also compares GraphQL with Falcor [2]. In our own work [21] we have studied how a REST-based backend can be migrated to a GraphQL server and examined the performance implications on different types of requests (i.e. consecutive REST calls vs. single GraphQL queries). In the following sections, we describe our experience with GraphQL in a real world project and provide a comparison of GraphQL with REST.

3 GraphQL

GraphQL is a strongly typed query language, which provides a flexible syntax for describing data requirements and interactions for building client applications [6]. It is not an implementation though, but represents a standard for developing GraphQL server solutions and use it on the client-side. A *GraphQL server* implements the language features and required characteristics defined in this specification [5]. The language itself provides a type system, which specifies the types and expressions that are supported. Each GraphQL server supports different types representing an application-specific type system. The possibilities of a GraphQL server are described by a *schema*, which specifies all supported types and operations. In the context of a given schema, a server can verify that a request is syntactically correct, unambiguous and mistake-free. As shown in Listing 1.1, the GraphQL schema always has one `schema` type that defines the entry point for all client operations.

Listing 1.1. Example of a GraphQL schema

```

schema {
  query : Query,
  mutation: Mutation,
  subscription: Subscription
}

type Query {
  maintenance(id: String): Maintenance
}

type Property {
  id : String!,
  customer: Customer
  ...
}

type Customer {
  id : String!,
  country: String!,
  city: String,
  ...
}

type Maintenance{
  id : String!,
  start: String!,
  property: Property
  ...
}
  
```

The following *operation types* can be defined in a schema:

Query. The *query type* is the entry point for each client-side request. Within this type, all possibilities for querying data are defined by the inclusion of further object types.

Mutation. If it is necessary to allow the clients to add or modify records, a *mutation type* can be defined as an entry point for respective requests.

Subscription. The *subscription type* offers an entry point for *real-time exchange* of data between clients and the server.

An *object type* represents a grouping of attributes that are referred to as *fields*. Each field contains a name, a type and an optional argument. It can also be specified, if attributes are non-nullable or arrays. The type is either a *scalar type* representing a single scalar value or an *object type*.

To issue a request, a *query document* must be created in a JSON-like syntax. A query document contains multiple definitions of *operations* and *fragments*. Each operation starts with an *operation type*. Depending on the schema, *query*, *mutation* and *subscription* might be available. In an operation (e.g. query), field arguments might be given to select objects and field names must be declared to be included in the results.

Fragments allow to reuse common repeated selections of fields. Fields can be either specified directly or the *spread operator* (...) can be used to include a fragment. All fields defined by a fragment will be added at the same level as the fragment invocation.

Listing 1.2 (left) presents an example of a request, where specific fields of the Maintenance-object with the ID of 19 are requested. As shown in Listing 1.2 (right), the server returns a JSON object with the requested data.

Listing 1.2. Example query (left) and result (right)

```
{
  maintenance(id: "19") {
    id
    ...
    Property {
      id
      ...
      Customer {
        id
        ...
      }
    }
  }
}

{
  "data": {
    "Maintenance": {
      "id": "19",
      "Property": {
        "id": "24",
        "Customer": {
          "id": "33"
        }
      }
    }
  }
}
```

The GraphQL specification offers more aspects of schema definition and the query language. For the sake of giving a first impression, we confine the discussion to the above selection of basic GraphQL features.

4 Migrating a Smart Home App from REST to GraphQL

This section presents the integration of GraphQL into the customer project *SmartHome* of diva-e Netpioneer GmbH. First we introduce the original architecture and present different approaches to integrate GraphQL. Then we show how a single service can be migrated to GraphQL and describe which adjustments had to be made. Subsequently, we present the results of an experimental evaluation that compares the performance before and after migration.

4.1 Original SmartHome System and Architecture

The SmartHome system provides a platform through which IoT devices can be managed. The backend architecture is organized in three layers for *web API*, *services* and *persistence*. Web API and services had been originally implemented as a Java application with Spring Boot¹. The API layer provides various RESTful services to request or transmit data by a number of different clients.

Listing 1.3 shows parts of the existing REST schema used in the performance analysis (Sect. 4.4). Each of the available resources in the system can be requested using a URI. For example, the resource *customer* can be requested by the URI `.../1.0/customer/{customerID}`. The data model of the REST API represents relationships between resources through primary keys (*id*).

¹ <https://projects.spring.io/spring-boot/>.

Listing 1.3. Example of the original REST data model

```

Maintenance {
    String id;
    Long start;
    Long end;
    String refProperty;
    ...
}

Property {
    String id;
    String street;
    String city;
    String comment;
    String refCustomer;
    ...
}

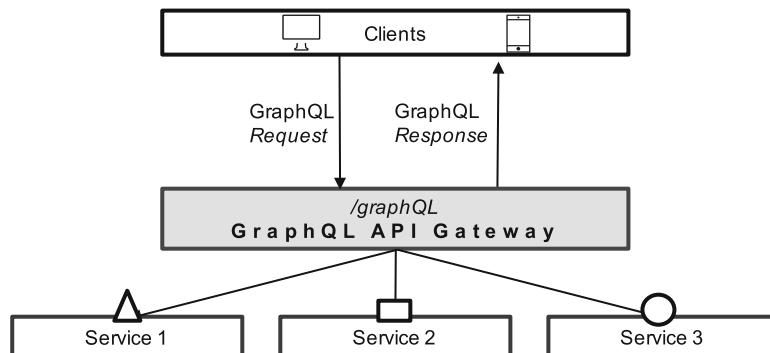
Customer {
    String id;
    String name;
    String street;
    String zip;
    String city;
    ...
}

```

In addition to an Angular single-page web app², various iOS and Android apps as well as IoT devices communicate with the server. In these applications, the views are often composed of several interdependent resources. For example, for a dashboard in the web app, the resources `Property`, `Maintenance`, and `Customer` shown in Listing 1.3 are required. Since the relationships require sequential resolution, several REST requests are necessary. Here, GraphQL is intended to improve the performance and flexibility of data exchange.

4.2 Architectural Considerations

In the following, we will discuss the integration of GraphQL. To use GraphQL, an interface is required, which is provided by a server. The server must be able to process GraphQL queries and return a data set specified by the client. As shown in Fig. 1, a possibility is to use GraphQL as an *API gateway* that can be used to encapsulate access to the external systems [9].

**Fig. 1.** GraphQL as an API gateway

This approach can be used in a microservice environment for encapsulating internal service interfaces and provide a unified web API. In a monolithic system, GraphQL is often directly integrated as a part of the API layer with direct access

² <https://angular.io>.

to intra-process services of the business layer. This approach has the advantage that no further network requests are necessary for a GraphQL query that could have negative effects on the performance.

4.3 Integration of GraphQL

A requirement for migration was the coexistence of original architectural layers and REST interfaces together with an additional GraphQL interface. For this reason GraphQL was integrated as part of the existing server. As described in Fig. 2, only the API layer had to be expanded.

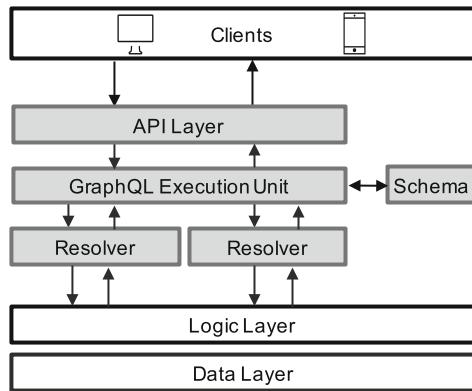


Fig. 2. GraphQL components in the migrated architecture

To this end, the *GraphQL endpoint* can be provided via Spring Boot and is (architecturally) located on the same level as existing REST interfaces. The GraphQL interface was created as a *POST interface*³ via URI path `/graphql`. The request payload contains a GraphQL query as specified by the client.

In order to implement the GraphQL specification, the open-source library `graphql-java`⁴ was used. The library provides three components that must be added to the existing system: the GraphQL *execution unit*, a *schema* and *resolvers*. As explained in Sect. 3, the schema describes how types can be queried or modified via the GraphQL interface. In the specified endpoint, a GraphQL query document is passed to a GraphQL execution unit where it is validated against the schema. Each object type in this schema is represented by a resolver. Resolver methods are responsible for getting the required data fields of the query or mutation from the underlying logic layer.

To achieve a homogeneous API the GraphQL schema was derived from the existing REST resources. In the GraphQL schema references between REST

³ See REST discussion in Sect. 5.1.

⁴ <https://github.com/graphql-java/graphql-java>.

resources were mapped to references between GraphQL object types. Excerpts of the resulting GraphQL schema are shown in Listing 1.1 (corresponding to the REST schema in Listing 1.3).

In the end, integration of GraphQL doesn't affect existing REST interfaces. Parallel operation of REST and GraphQL is possible without restrictions.

4.4 Performance Analysis

In order to evaluate the envisioned performance gains of the GraphQL API compared to the original REST API, we have conducted performance tests. In the following, results of two experiments are presented, each of which consisting of 1000 requests. Results are shown as box plot diagrams (boxes represent 50% of results; upper/lower ends show max/min values, dotted lines represent medians).

In the first test scenario, an atomic resource `customer` was retrieved with three different approaches: 1. request against single REST API endpoint, 2. regular GraphQL query, 3. reduced GraphQL query. In the case of the reduced query, only a single field was queried instead of all fields to determine differences in query size. For all three retrieval approaches, one roundtrip was necessary to fetch the data record. Figure 3 shows the resulting box plot of experiment 1.

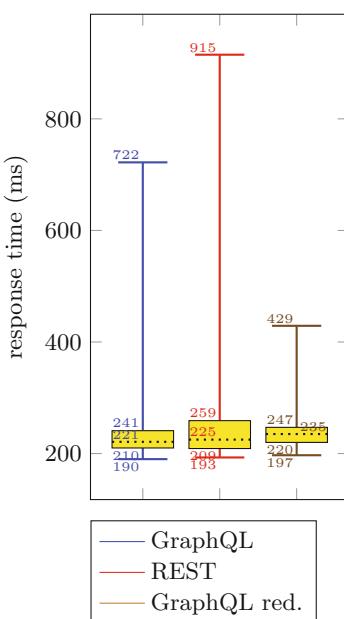


Fig. 3. Experiment 1: retrieving atomic resources

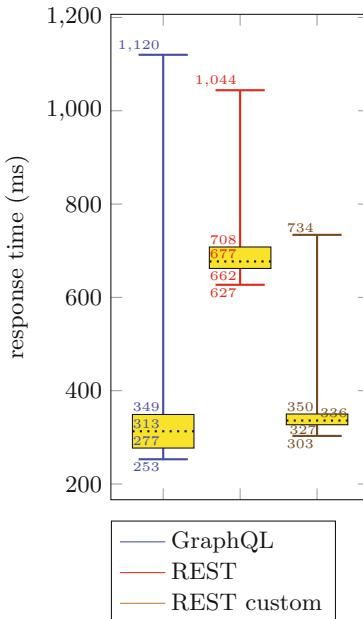


Fig. 4. Experiment 2: retrieving multiple linked resources

The median was 221 ms for performing the GraphQL query, and 225 ms for a request using the existing REST API. Thus, there is no significant difference

between the median of both retrieval approaches. The average query time of the reduced GraphQL query was 235 ms, so no significant difference could be found in the response times either. From this experiment we conclude that *GraphQL is not slower regarding query execution than REST for atomic resources*.

In experiment 2, the resources `Property`, `Maintenance`, and `Customer` depict in Listing 1.3 were retrieved requiring traversal of the original resource graph. The GraphQL approach uses a single query (shown in Listing 1.2) to request all objects from the server. The second variant utilizes the existing REST API to retrieve three resources consecutively one after the other. For the third variant, an optimized REST endpoint was created specifically for this application, which returns the resources in a single response. Figure 4 shows the resulting box plot of experiment 2.

The median was 313 ms for the GraphQL query and 677 ms for consecutive requests using the original REST API. For calling the specific endpoint, the median was 336 ms. Therefore, GraphQL required 46% of the time on average to retrieve all data from the server compared to the REST API. Response times of the specific endpoint were in a similar range than that of GraphQL.

5 Challenges of Migration

In the previous section, we have given a concrete example of migrating a static REST/HTTP API to GraphQL. It shows that the performance of client-server communication can be significantly improved. However, trade-offs had to be made for the migration and some issues have not been considered.

Challenges are caused by conceptual differences between REST and GraphQL on architectural level. These translate to concrete technical issues when creating individual services. Both angles are being discussed subsequently.

5.1 Challenges on Architectural Level

REST is an *architectural style* with general principles resulting in beneficial characteristics on system-level like flexibility, scalability and robustness. It is often interpreted as an implementation based on web standards (HTTP, URI, XML/JSON) leading to systems, which are simple, cheap and interoperable.

When migrating a REST/HTTP API to GraphQL/HTTP, a general challenge is not to break the design principles of the original architecture for the above reasons. Based on the four principles given by Pautasso et al. [19], we therefore analyze the compatibility of GraphQL with REST.

Resource Identification through URIs. REST encourages versatile resource-oriented architecture where self-contained cohesive resources are individually addressable by means of URIs. GraphQL on the other hand promotes a more data-centric model without architectural resources. A GraphQL service represents an *object graph* of data entities. These objects are collectively accessible through a single endpoint and URI.

Uniform Interface. REST APIs take full advantage of HTTP verbs and semantics (at least GET and POST should be considered for distinguishing idempotent operations). GraphQL introduces a high level query protocol on top of HTTP with individual operations (*query*, *mutation*, *subscription*). GraphQL requests might be alternatively mapped to GET or POST operations regardless of HTTP semantics.

Self-Descriptive Messages. REST promotes HTTP content negotiation, usually in conjunction with common formats like XML, JSON or HTML and HTTP metadata for resource representation. GraphQL prescribes a fixed query language for requests but is agnostic to the response (serialization) format (JSON is commonly used). Still the logical structure of a response is restricted by the specification and mirrors the intent of GraphQL as a data-fetching technology.

Stateful Interactions through Hyperlinks. Interactions with single REST resources are stateless and require self-contained messages. Conversation state is advanced by following hyperlinks that are transferred as part of resource representations in the former response. GraphQL interactions are stateless as well but are usually not meant as part of a stateful conversation. Traversal of the object graph happens during execution of a single query for the sake of collecting individual result sets.

From the above discussion it becomes obvious that there are severe conceptual differences between the general models and practical realizations of REST and GraphQL. Consequently, a *substitution* of former REST resources by means of GraphQL services is likely to break the design principles of the original architecture. In particular, substituting the entire resource graph by a monolithic GraphQL service would be clearly invasive.

A more promising migration approach is to complement the original resource graph by additional endpoints representing subsets of linked resources. Still such complementary GraphQL services should be carefully designed, i.e. with respect to the selection of appropriate resources that lend themselves to data fetching as well as their granularity.

5.2 Challenges on Service-Level

Conceptual differences of GraphQL compared to the original REST architecture shows in a number of issues with respect to the implementation of GraphQL service endpoints. Degeneration of the HTTP-level as regards addressing of individual resources or semantics of operations leads to typical problems of utilizing common techniques for caching, fault tolerance or authorization. Some additional issues emerge on the level of the GraphQL protocol like new security threads. In the following, we will focus on caching and DoS threads.

Caching. GraphQL objects are not addressable via URIs. Thus, caching behavior of GraphQL queries cannot be specified by HTTP headers and generic HTTP

caching can't be applied. Instead, we need specific GraphQL caching that utilizes *object IDs* within queries. It is important to note that *globally unique IDs* are required for this purpose [6].

We might implement dynamic content caching by means of various approaches, depending on query and update behavior [20]. Full or partial *replication of data* improves situations with complex queries and infrequent updates. *Content-blind caching* hashes normalized queries as result keys. It works well for repeated queries but might pollute the cache otherwise. This approach works on HTTP-level, if GraphQL requests use HTTP GET with queries encoded as URL query parameters. *Content-aware caching* evaluates queries and locally stores resulting objects for a limited set of query templates. It is a trade-off between cache size and range of supported queries but puts more load on the cache for query evaluation and requires more involved consistency mechanisms.

As regards placement of cache content, it might be put on the GraphQL server (to avoid accessing remote data stores), dedicated proxies or the client-side. For the case of web apps, GraphQL clients such as Apollo⁵ or Relay⁶ already offer browser-located caching out of the box.

Denial of Service Attacks. GraphQL enables clients to submit individual queries to the server for execution. This feature might be exploited by perpetrators generating excessive load through overly complex (single) queries possibly leading to service failure. Therefore, we need to introduce countermeasures securing GraphQL APIs against malicious queries.

A naive approach against excessive requests is to check the *size of the query representation* before execution and restrict it to a certain limit. The GraphQL feature of *persistent queries* offers another simple way of protection. Here, all permitted query templates are stored on the server. Clients might only execute pre-defined queries, which can certainly be seen as benefit and drawback at the same time.

More sophisticated approaches introduce cost metrics quantifying the effort of query execution. *Resolver restriction* builds on the weighted number of resolver calls for a query or, alternatively, the duration of resolver execution. If a request exceeds some given threshold, the resolver interrupts it. Obviously, harmless queries might be killed and malicious ones still run for some time.

A better way would be to statically analyze queries prior to execution and disregard them in case of excessive complexity. Hartig and Pérez have shown that queries might lead to resulting object graphs of exponential size [10]. Luckily they also suggest that the size of a result object can be estimated with realistically low complexity. This would provide the theoretic foundation for effective DoS prevention to be built into GraphQL server frameworks.

⁵ <https://github.com/apollographql/apollo-client>.

⁶ <https://facebook.github.io/relay/>.

6 Conclusion and Outlook

In this paper we have studied the migration of distributed (e.g. web-based) apps from RESTful client-server interactions to GraphQL. We have compared the performance of complementary web APIs in the context of an industrial case study and demonstrated the benefits of the data fetching approach with GraphQL. While we restrict the study to a minimal part of the API, we consider the behavior of the real world backend (business and persistence layers) as representative for the performance measures.

Integrating complex semi-compliant technologies in the context of distributed systems naturally introduces challenges. Even the individual technologies themselves usually leave room for interpretation beyond their given rules or specification. At this point, best practices need to fill these gaps.

If generic solutions are found, *patterns* can be deduced. Some patterns have already emerged for GraphQL like *Connections* for pagination or *Globally Unique Object IDs* for caching. We expect more patterns to emerge in the architectural space and related to the integration of GraphQL with REST/HTTP. The ultimate goal would be to provide a systematically structured set of patterns as part of a more general pattern language for web APIs.

References

1. Byron, L.: GraphQL: A data query language. <https://code.facebook.com/posts/1691455094417024/graphql-a-data-query-language/>. Accessed 09 June 2017
2. Cederlund, M.: Performance of frameworks for declarative data fetching: an evaluation of Falcor and Relay+GraphQL. Master's thesis, KTH, School of Information and Communication Technology (ICT), Stockholm, Sweden (2016)
3. Cupek, R., Huczala, L.: OData for service-oriented business applications: Comparative analysis of communication technologies for flexible Service-Oriented IT architectures. In: 2015 IEEE International Conference on Industrial Technology (ICIT), Seville, pp. 1538–1543. IEEE (2015)
4. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Pearson Education, Upper Saddle River (2005)
5. Facebook Inc.: GraphQL - Working Draft. <http://facebook.github.io/graphql/>. Accessed 11 Apr 2017
6. Facebook Inc.: Introduction to GraphQL. <http://graphql.org/learn/>. Accessed 19 Apr 2017
7. Feuerlicht, G., Lozina, J.: Understanding service reusability. In: 15th International Conference Systems Integration, Department of Information Technologies and Czech Society for Systems Integration, Prague, Czech Republic, pp. 144–150 (2007)
8. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
9. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
10. Hartig, O., Perez, J.: An initial analysis of Facebook's GraphQL language. In: Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW), Montevideo, Uruguay, 5–9 June (2017)

11. Helfer, J.: GraphQL - Evolution or Revolution? <https://speakerdeck.com/helper/graphql-evolution-or-revolution>. Accessed 27 June 2017
12. Helfer, J.: GraphQL vs. Falcor. <https://dev-blog.apollodata.com/graphql-vs-falcor-4f1e9cbf7504>. Accessed 17 July 2017
13. Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTR (2004)
14. Lewis, J., Fowler, M.: Microservices. <https://martinfowler.com/articles/microservices.html>. Accessed 17 July 2017
15. Meredith, C.: The Falcor data model is a graph, and the GraphQL data model is a tree. <https://edgecoders.com/the-falcor-data-model-is-a-graph-and-the-graphql-data-model-is-a-tree-6748ba53bb96>. Accessed 03 Aug 2017
16. Miller, D.: graphql (facebook), falcor (netflix) and odata and ... <http://appddeeovvmeanderings.blogspot.com/2016/02/graphql-facebook-falcor-netflix-and.html>. Accessed 17 July 2017
17. Netflix Inc.: Falcor: One Model Everywhere. <https://netflix.github.io/falcor/>. Accessed 17 July 2017
18. OData: OData - the Best Way to REST. <http://www.odata.org/>. Accessed 17 July 2017
19. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. “big” web services: making the right architectural decision. In: Proceedings of the 17th international conference on World Wide Web, pp. 805–814. ACM (2008)
20. Tanenbaum, A., Van Steen, M.: Distributed Systems: Pearson New International Edition: Principles and Paradigms. Pearson Education Limited, Upper Saddle River (2013)
21. Vogel, M.: Potential von GraphQL in dynamischen Webanwendungen. Bachelor’s Thesis, Karlsruhe University of Applied Sciences, Karlsruhe, Germany (2017)



Using Risk Patterns to Identify Violations of Data Protection Policies in Cloud Systems

Stefan Schoenen^(✉), Zoltán Ádám Mann^(✉), and Andreas Metzger^(✉)

Paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen,
Essen, Germany

{Stefan.Schoenen,Zoltan.Mann,Andreas.Metzger}@paluno.uni-due.de

Abstract. Cloud services and cloud infrastructures become increasingly complex and dynamic: many different physical and virtual machines, applications and their components interact and all of these entities may be differently reconfigured, deployed, and migrated during run time. In addition, a multitude of stakeholders may be involved in cloud service offering and usage; e.g., service consumers, cloud providers, data subjects, data controllers, and actual end users. Thus, checking whether cloud services comply with data protection policies when storing or processing sensitive data becomes a challenge due to the involved complexity and dynamicity. We present a model-based approach for identifying violations of data protection policies at run-time. Key elements of our approach are (1) a run-time model to represent the actual cloud system and its stakeholders at runtime, and (2) risk patterns that commonly appear in the context of data protection issues. Our approach aims to find instances of these risk patterns in the run-time model. If an instance of a risk pattern is found, this indicates a risk of data protection violation. We demonstrate the applicability of our approach by using an industry scenario.

Keywords: Cloud computing · Data protection · Privacy
Run-time model · Risk pattern

1 Introduction

The compelling advantages of cloud computing, such as the seemingly infinite resource provisioning without the need for buying costly IT equipment, have made the cloud the platform of choice in many domains [5]. However, using the cloud is often associated with a loss of control since multiple parties – e.g., operators of cloud services – may potentially have access to data and code of applications in the cloud. Thus, storing and processing sensitive data in the cloud poses additional risks, hampering the adoption of the cloud [27].

There are several well-known techniques to ensure confidentiality and integrity of data in a cloud environment, e.g., encryption and authentication

technologies. However, they all have some drawback, e.g., in terms of performance overhead (homomorphic encryption) or inconvenience for users (multi-factor authentication). Therefore, deciding which mechanism(s) to use is not trivial and thus a certain knowledge base about when to use which is required.

Traditionally, this decision was made during design by experienced software engineers. However, to support agile deployment processes in a DevOps environment and to leverage the potential of the cloud in terms of flexibility, decisions on data protection mechanisms to use have to be made increasingly at deployment time or at run time. For example, an application may need only light-weight data protection mechanisms if deployed in a private cloud, but full-fledged data protection if deployed in a public cloud. During design time, it may not be clear where the application will be deployed [4, 28], so the application must be engineered such that the data protection mechanisms for the public cloud scenario are available but can be turned off if the application ends up being deployed in a private cloud. The mechanisms will then be turned on or off at deployment time. Moreover, the application may be migrated during run time between clouds using live migration [16], in which case data protection mechanisms may need to be turned on or off even during run time.

To enable this flexibility, *decisions on the use of data protection mechanisms must be made automatically* during deployment and run time, of course within bounds set during design time. Thus, the task of software and service engineers changes from making a concrete design decision to *providing the decision logic and determining the information on which the decision should be based*. In particular, this calls for real-time risk identification and assessment that can be used to trigger the appropriate mitigation actions during run time.

In this paper, we focus on the question of *what information is needed* to identify risks of data protection violations. Specifically, we argue that this involves the following pieces of information:

- A model of the – current or planned – configuration of the relevant *assets*, including infrastructure elements, middleware, applications, and data, but also involved actors;
- A set of *risk patterns*, which describe asset configurations that would cause too high risks of data protection violation and hence must be avoided.

The main novelty of our approach is its holistic view in two dimensions: considering (i) all layers of the cloud stack, (ii) from design time to run time. This view is needed to make sound decisions, because data protection is a cross-cutting concern and the required knowledge is established partly at design time but partly only when the system is deployed or adapted [17].

The remainder of the paper is organized as follows: Sect. 2 presents a typical cloud deployment scenario that was created with industry partners to illustrate the potential data protection issues. Sect. 3 gives an overview of our approach for identifying violations of data protection policies. The details are described in Sects. 4, 5 and 6. It is applied to the scenario from Sect. 2 in Sect. 7. Related work is discussed in Sect. 8. The paper is concluded in Sect. 9.

2 A Motivating Example

In this section, we look at an industrial cloud setup and its implications on data protection. This scenario has been devised in the context of the project “RestAssured – Secure Data Processing in the Cloud”¹ with several industry partners. While it abstracts from specific applications, it has been validated to reflect the typical data protection concerns of practical cloud systems.

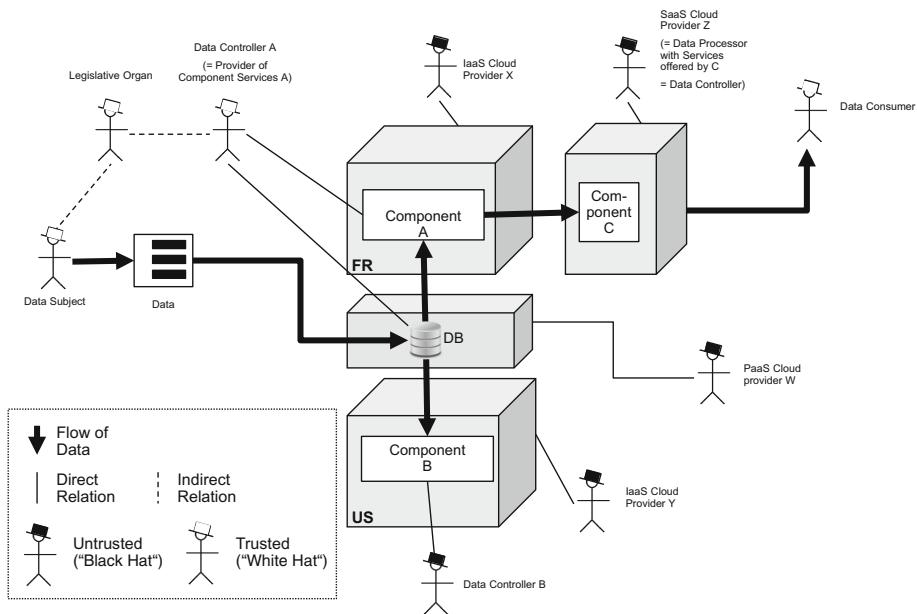


Fig. 1. An industrial cloud scenario

In Fig. 1 we see a typical cloud scenario where multiple parties are involved. In this scenario, personal, unencrypted data (a record) about individual users (Data Subject) are captured by a company (Data Controller A), with explicit consent of the users and under legislative control. The company stores the data in an unencrypted database (DB) operated by a PaaS (Platform as a Service) provider W and deploys its application (Component A) using the infrastructure (including a VM in which Component A runs and a PM in which this VM runs) provided by IaaS (Infrastructure as a Service) provider X. Another actor (Data Consumer) uses an application (Component C) to communicate with Component A and get access to the data. Component C is run by SaaS (Software as a Service) provider Z. Another company (Data Controller B) uses an application (Component B) run on the infrastructure of IaaS provider Y that accesses DB.

¹ <https://restassureddh2020.eu/>.

It is important to note that Data Controller A and Data Consumer (white hats) are trusted by Data Subject. Nevertheless, as the data traverse between the trusted parties, several untrusted actors (PaaS provider W, IaaS provider X, SaaS provider Z) may get unauthorized access to the data along the way. Moreover, other cloud tenants using the same public database offering (DB) can also get access to the data. The latter also poses a new problem because Component B is hosted in the US, but European regulations prohibit processing personal data of EU citizens outside the EU.

As we can see, a cloud setup can be complex with many different socio-technical interactions, posing a wide-ranging set of threats to data protection.

3 Overview of the Proposed Approach

To address the challenges of ensuring data protection for dynamically deployed and configured cloud services, we devise a model-based approach spanning activities from design time to run time. Figure 2 shows an overview of our approach.

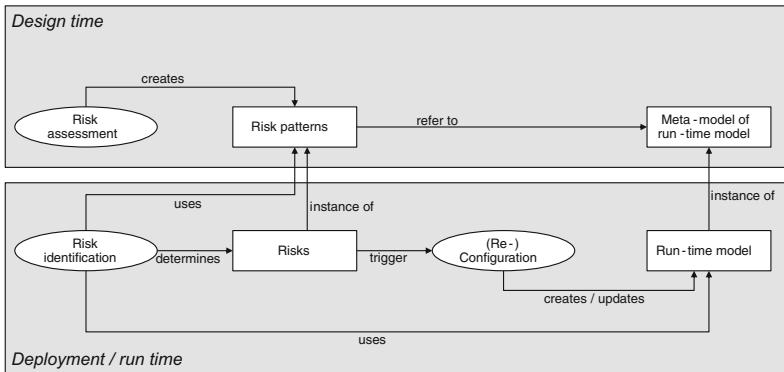


Fig. 2. Overview of our approach. Boxes represent artefacts, ovals represent activities.

Our approach revolves around two types of artefacts: Run-time model and risk patterns. The run-time model (see Sect. 4) is created at deployment time and kept updated during run time, so that it always reflects the current configuration of the cloud-based application and its environment. The risk patterns (see Sect. 5) are created at design time to capture situations that must be avoided because of the associated high risk of data protection violations. These two artefacts are used by the “Risk identification” activity during deployment time and run time to identify risks of data protection violation (see Sect. 6).

4 Run-Time Model

Figure 3 shows an overview of the suggested run-time (meta-)model. It is based on models proposed previously in the literature [10, 26], but extends them to have all necessary information for the identification of risks to data protection.

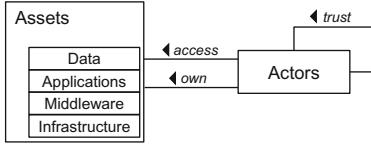


Fig. 3. Overview of the meta-model for run-time models

The run-time model consists of two main parts:

- The *Assets* part allows capturing the current cloud configuration, thereby providing a basis for reasoning about possible data protection violations.
- The *Actors* part allows capturing the relevant actors (natural persons as well as organizations), their roles, attributes, and relations, thereby allowing to reason about their data protection requirements.

These two parts are described in more detail next.

Assets. The most important assets are the data that we want to protect. But other types of assets are also important because they may provide additional attack surfaces (see Sect. 5 for an example). So it is important to consider other asset types as well – here: Applications, Middleware, and Infrastructure.

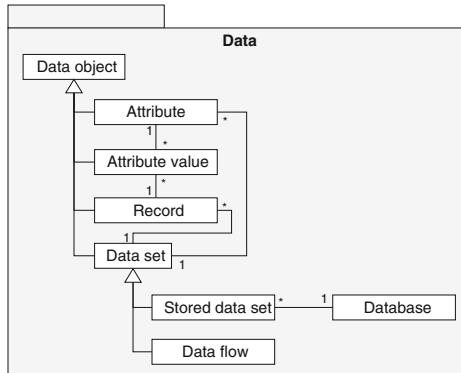


Fig. 4. A possible refinement of the “Data” part of the meta-model for run-time models

The Data, Applications, Middleware, and Infrastructure parts of the run-time model contain the entities that make up the given layer of the cloud stack, along with their attributes and relations. As an example, Fig. 4 shows a possible set of entity types and their relations in the Data part of the meta-model. It should be noted that this is just an example; the exact types may vary depending on the specifics of the used data model (e.g., relational or not).

The Applications part of the model contains information about the structure of the application in terms of components and connectors, the Middleware part contains the entities modeling standard software platform components like database servers and application servers, whereas the Infrastructure part models the underlying physical and virtual resources.

The Assets model must also contain the relations between entities in different parts of the model, for example a data flow that passes through a connector of two application components.

Actors. For modeling the actors, we consider two different kinds of roles:

- Roles related to cloud services. For all involved services (IaaS, PaaS, SaaS), we can differentiate between developers, operators, and users of the services.
- Roles related to data protection. In accordance with the EU’s General Data Protection Regulation [7], we use the roles data subject, data controller, and data processor.

Of course, it is possible that multiple roles belong to the same actor. For example, an organization can be user of an IaaS service and operator of an SaaS service.

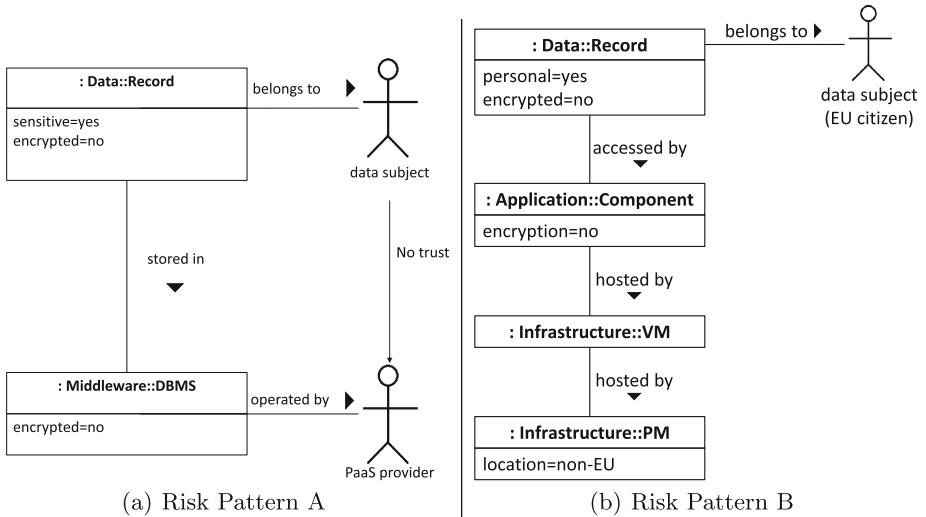
In terms of the relations among actors, *trust* is of special importance. We use a white-list approach to trust, i.e., every trust relation must be explicitly established (e.g., by means of a contract). Also, trust relations can be limited to specific types of actions on specific data.

5 Risk Patterns

Risk patterns are the core concept in our approach for identifying potential data protection violations. A risk pattern describes a configuration of assets and actors that would lead to unacceptably high risk of data protection violations and hence must be avoided. Syntactically, risk patterns are expressed in terms of the entities that make up the meta-model of the run-time model, their attributes and relations.

Figure 5 shows two examples for risk patterns. In the example of Fig. 5(a), there is a sensitive data record stored in a database operated by a PaaS provider that is not trusted by the data subject. Neither the data record nor the database is encrypted, so the PaaS provider could access the sensitive data, leading to a possible data protection violation. In the example of Fig. 5(b), a personal data record is accessed by an application component that is hosted by a VM in a PM in a non-EU location. Since personal data of EU citizens must not be processed outside the EU, this would also lead to a data protection violation.

To clarify the semantics of risk patterns, it should be noted that the objects in the risk patterns are not specific entities, but should be considered as variables that can take on any specific object of the type as value. The specification of attributes and relations in the risk pattern is to be considered as constraints on these attributes and relations. The statement expressed by a risk pattern is

**Fig. 5.** Two sample risk patterns

that, whenever the variables can be instantiated with specific objects from the run-time model satisfying the given constraints on attributes or relations, this represents an unacceptable case. In the example of Fig. 5(a), the statement can be formulated as follows: it is unacceptable if there is a data subject, a data record, a DBMS and a PaaS provider such that the data record belongs to the data subject, is sensitive and not encrypted and stored in the DBMS, the DBMS is not encrypted and operated by the PaaS provider, and the data subject does not trust the PaaS provider.

The examples show that risk patterns can express complex situations in a compact way. They can adequately capture data protection issues that arise through *interactions of multiple actors and entities on different cloud layers*, also taking into account their attributes and relations.

6 Activities from Design Time to Run Time

Now that we have described the pieces of information forming the core of the suggested approach, we also briefly describe the activities shown in Fig. 2.

Design time. At design time, the meta-model of the run-time model is defined. This also determines the language that is used to express the risk patterns.

Risk patterns are derived from a risk assessment. The risk patterns are designed as a black-list: Each unwanted situation must be captured by a risk pattern. For determining risk patterns, a risk analysis process should be followed, focusing on the types of assets, vulnerabilities and threats in the system. Risks are identified and assessed in terms of their probability and impact. There are well-known methodologies and standards for such a process [12, 18].

Deployment time. When the system is deployed in the cloud, the run-time model is created based on the target environment and the planned configuration of the application. When the run-time model is in place, the risk patterns can be evaluated for the first time to check whether the configuration would lead to any data protection issues. The evaluation of the risk patterns can be cast as a graph pattern matching problem, in which one tries to find a subgraph of the run-time model that matches the risk pattern. For this, graph pattern matching algorithms can be used [6, 15].

If a match was found, i.e., a risk was identified, the configuration is changed and checked again until a safe configuration is found.

Run time. During run time, the system can use self-adaptation to react to changes in its environment using the MAPE model [14]. Monitoring is used to detect relevant changes and update the run-time model accordingly. For this purpose, existing cloud monitoring tools and further instrumentation can be used. The impact of observed changes is analyzed: the same evaluation logic as during deployment can be used to detect risk pattern matchings in the run-time model. If necessary, the same configuration logic as during deployment can be used to devise a new, safe configuration of the system (planning). Finally, the changes are executed by reconfiguring the system accordingly. The reconfiguration can be done either automatically or using operator-in-the-loop adaptation [13].

It should be noted that adaptation can be also caused by other reason (e.g., insufficient performance). Then, the pattern-matching logic can be used to ensure that the new configuration of the system fulfills data protection requirements.

7 Application to Our Cloud Scenario

In this section we revisit our example cloud scenario from Sect. 2. Figure 6 depicts an excerpt from the corresponding run-time model expressed as an instantiation of the meta-model from Sect. 4. The figure also shows the two matched sample risk patterns from Sect. 5 (framed with dashed lines), as they could be found by a graph pattern matching algorithm:

- Risk Pattern A led to detecting the risk that an untrusted PaaS provider can access a confidential data set stored in the provider’s database.
- Risk Pattern B led to detecting the risk associated with the data subject’s data being processed on a physical machine outside the EU (here in the USA).

After identifying these specific risks, the configuration of the system can be adapted in such a way that the found risks are mitigated, which will be established by the pattern matching algorithm not being able to find a match with the defined risk patterns. Hence, the run-time model and the risk patterns together can indeed be used to detect violations of data protection policies and – by using the system reconfiguration during runtime – ultimately to avoid them.

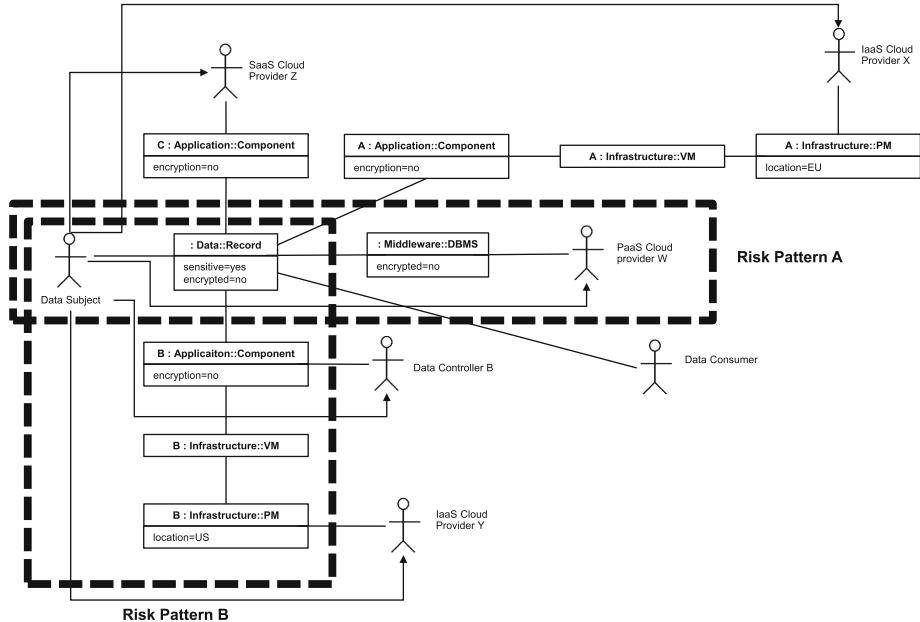


Fig. 6. Run-time model of the example from Sect. 2. The found risk pattern instances are encompassed by the dashed lines.

8 Related Work

We discuss the work most relevant to our proposed approach along the aspects (1) monitoring data protection concerns in the cloud, (2) risk management addressing data protection concerns, and (3) data protection risks of cloud services.

Data Protection Monitoring for Cloud Services. While much work on cloud and service monitoring (e.g., see [1, 23]) has been published, approaches for data protection compliance monitoring of services are scarce [20].

Alhamazani et al. [2] discussed the design issues of several monitoring tools and also those of cloud monitoring in general. However, data protection is only mentioned as a minor aspect of monitoring in comparison to performance or QoS regulations. Our approach is mainly focused on data protection.

Foster and Spanoudakis [11] proposed a monitoring approach for cloud services which takes data protection aspects into consideration. Their approach covers only authenticity and auditability as parts of security and ignore other security goals like confidentiality and integrity. Our approach can be used to cover these security goals as well.

Nikolai and Wang propose an IaaS intrusion monitoring and classification system [21]. This system has the purposes of reducing the overload from security sensors by detecting only anomalies which are classified as “urgent” and classifying the attack which causes abnormal behaviour. Our approach classifies

these security risks depending if a risk pattern was detected or not. Besides IaaS we also take PaaS and SaaS into consideration.

Watson and Little [28] introduce an approach to reason about the deployment of a distributed system and its impact on security. They state that not all deployment problems can be solved during design time, so run time reasoning is needed. In contrast to our risk patterns, their approach requires the assignment of security levels to all assets, which can be difficult in some settings. Beyond the types of entities considered in that paper, we also explicitly consider actors. As shown in our paper, actors are important for accurately determining data protection concerns.

Meziane et al. [20] describe a monitoring approach for identifying violations with respect to data usage. They introduce privacy-aware SLAs against which data usage flows are checked during run time. Our risk pattern approach takes a broader stance and also covers other aspects relevant for data protection, such as secure storage or processing of data.

In our own previous work, we have introduced a run-time model-based approach for detecting geo-location policy violations [24, 25]. Even though this previous work also relies on run-time models, it only considers geo-location as an aspect of data protection, e.g., data may not be migrated outside of the EU. Geo-location is only one relevant aspect for data protection, and so the risk-based approach we introduce here has much broader scope and applicability.

Data Protection Risk Management for Cloud Services. Risk management covers the process of describing, detecting and mitigating risks. So far, only few frameworks for risk management of services have been presented [19].

Djemame et al. [8] propose a risk assessment framework for cloud computing which is designed to help cloud users and cloud providers assess risks during service deployment and operation. This approach focuses on the relationship between service providers and services. However, they do not state how risks may be monitored during operations. This is where risk patterns can help.

Meszaros and Buchalcevova [19] present a framework for online service risk management. They consider similar assets to ours and present a risk and threat model as basis. They focus on risk assessment and mitigation and propose techniques for risk monitoring. Our approach can be considered complementary to their work as we consider the impact of reconfiguration of cloud services on data protection risks.

Data Protection Risks of Cloud Services. Several authors have analyzed specific data protection risks in the context of cloud computing and services. Paquette et al. [22] analyzed the risks of cloud computing, focusing on the context of governmental use of cloud computing. Ardagna et al. [3] survey data-protection-related publications, stating that the areas where risks occur are on application-level, between multiple tenants, or between a provider and a tenant. Fernandes et al. [9] survey security issues in cloud computing as a potential source for data protection risks. These insights provide an important source of input for our approach as they help defining and specifying risk patterns by

taking important data protection concerns into account. Our approach can be seen as a vehicle for capturing and utilizing this kind of knowledge.

9 Conclusion and Future Work

In this paper, we proposed an approach for identifying potential data protection violations in cloud systems. The approach focuses on a run-time model of relevant cloud entities and a set of risk patterns to capture situations that would lead to unacceptably high risks. The identification of potential data protection violations is done using graph pattern matching during deployment time and run time.

The next steps of our research include the formalization of the concept of risk patterns and the adoption of efficient algorithms for the graph pattern matching problem. As a result, we hope to get a better understanding for the possibilities, limitations, and efforts relating to the proposed approach.

Acknowledgments. This work received funding from the European Union's Horizon 2020 research and innovation programme under grant 731678 (RestAssured). Useful discussions with project partners are gratefully acknowledged.

References

1. Aceto, G., Botta, A., de Donato, W., Pescapè, A.: Cloud monitoring: a survey. *Comput. Netw.* **57**(9), 2093–2115 (2013)
2. Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F.A., Jayaraman, P.P., Khan, S.U., Guabtni, A., Bhatnagar, V.: An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing* **97**(4), 357–377 (2015)
3. Ardagna, C.A., Asal, R., Damiani, E., Vu, Q.H.: From security to assurance in the cloud: a survey. *ACM Comput. Surv.* **48**(1), 2:1–2:50 (2015)
4. Brogi, A., et al.: SeaClouds: an open reference architecture for multi-cloud governance. In: Tekinerdogan, B., Zdun, U., Babar, A. (eds.) *ECSA 2016. LNCS*, vol. 9839, pp. 334–338. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48992-6_25
5. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **25**(6), 599–616 (2009)
6. Cheng, J., Yu, J.X., Ding, B., Philip, S.Y., Wang, H.: Fast graph pattern matching. In: *IEEE 24th International Conference on Data Engineering*, pp. 913–922 (2008)
7. Council of the European Union: General Data Protection Regulation (2016)
8. Djemame, K., Armstrong, D., Guitart, J., Macias, M.: A risk assessment framework for cloud computing. *IEEE Trans. Cloud Comput.* **4**(3), 265–278 (2016)
9. Fernandes, D.A.B., Soares, L.F.B., Gomes, J.V.P., Freire, M.M., Inácio, P.R.M.: Security issues in cloud environments: a survey. *Int. J. Inf. Sec.* **13**(2), 113–170 (2014)
10. Ferry, N., Rossini, A., Chauvel, F., Morin, B., Solberg, A.: Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In: *IEEE 6th International Conference on Cloud Computing*, pp. 887–894 (2013)

11. Foster, H., Spanoudakis, G.: Advanced service monitoring configurations with SLA decomposition and selection. In: Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), pp. 1582–1589 (2011)
12. Furuncu, E., Sogukpinar, I.: Scalable risk assessment method for cloud computing using game theory (CCRAM). *Comput. Stand. Interfaces* **38**, 44–50 (2015)
13. Heinrich, R., Jung, R., Schmieders, E., Metzger, A., Hasselbring, W., Reussner, R., Pohl, K.: Architectural run-time models for operator-in-the-loop adaptation of cloud applications. In: 9th Symposium on the Maintenance and Evolution of Service-Oriented Systems and Cloud-Based Environments (2015)
14. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* **36**(1), 41–50 (2003)
15. Mann, Z.A.: Optimization in Computer Engineering - Theory and Applications. Scientific Research Publishing (2011)
16. Mann, Z.A.: Approximability of virtual machine allocation: much harder than bin packing. In: Proceedings of the 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications, pp. 21–30 (2015)
17. Mann, Z.A., Metzger, A.: Optimized cloud deployment of multi-tenant software considering data protection concerns. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 609–618 (2017)
18. Martens, B., Teuteberg, F.: Decision-making in cloud computing environments: a cost and risk based approach. *Inf. Syst. Front.* **14**(4), 871–893 (2012)
19. Meszaros, J., Buchalcevova, A.: Introducing OSSF: a framework for online service cybersecurity risk management. *Comput. Secur.* **65**, 300–313 (2017)
20. Meziane, H., Benbernou, S., Hacid, M., Malik, Z., Papazoglou, M.P.: A view-based monitoring for usage control in web services. *Distrib. Parallel Databases* **34**(2), 145–178 (2016)
21. Nikolai, J., Wang, Y.: A streaming intrusion monitoring and classification system for IaaS cloud. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pp. 632–639, June 2016
22. Paquette, S., Jaeger, P.T., Wilson, S.C.: Identifying the security risks associated with governmental use of cloud computing. *Gov. Info. Q.* **27**(3), 245–253 (2010)
23. Rabiser, R., Guinea, S., Vierhauser, M., Baresi, L., Grünbacher, P.: A comparison framework for runtime monitoring approaches. *J. Syst. Softw.* **125**, 309–321 (2017)
24. Schmieders, E., Metzger, A., Pohl, K.: A runtime model approach for data geolocation checks of cloud services. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 306–320. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45391-9_21
25. Schmieders, E., Metzger, A., Pohl, K.: Runtime model-based privacy checks of big data cloud services. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 71–86. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48616-0_5
26. Shao, J., Wei, H., Wang, Q., Mei, H.: A runtime model based monitoring approach for cloud. In: 2010 IEEE 3rd international conference on Cloud Computing (CLOUD), pp. 313–320. IEEE (2010)
27. Tang, J., Cui, Y., Li, Q., Ren, K., Liu, J., Buyya, R.: Ensuring security and privacy preservation for cloud data services. *ACM Comput. Surv.* **49**(1), 1–31 (2016). art. 13
28. Watson, P., Little, M.: Multi-level security for deploying distributed applications on clouds, devices and things. In: IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 380–385 (2014)



Towards Setting Up a Collaborative Environment to Support Collaborative Business Processes and Services with Social Interactions

Andrea Delgado^(✉), Laura González, and Daniel Calegari

Instituto de Computación, Facultad de Ingeniería, Universidad de la República,
11300 Montevideo, Uruguay

{adelgado, lauragon, dcalegar}@fing.edu.uy

Abstract. In the last decade, the conjunction of Business Process Management Systems (BPMS) with Service Oriented Architecture (SOA) proposals has gained many adepts both in industry and academy, as the straightforward way to connect Business Processes (BPs) with the services that implement them. Nowadays collaborative and virtual organizations need increasingly support to enact their collaborative BPs, both in centralized and decentralized scenarios. The complexity of existing systems and the variety of languages and technologies available, are key elements for the use of services allowing their seamlessly integration. In this paper we present a proposal for setting up a collaborative environment comprising: (i) a reference architecture for a process-aware inter-organizational service integration platform (PA-IOSIP) defined in previous work, based on a BPMS platform and middleware infrastructure (e.g. Enterprise Service Bus (ESB)), and (ii) a maturity model which provides a roadmap to guide the efforts towards setting up such a collaborative environment. We also present a proof of concept we have carried out within the Uruguayan e-Government context.

Keywords: Collaborative Business Processes
Service Oriented Computing · Business Process Management Systems
Enterprise Service Bus

1 Introduction

In the last decade, the conjunction of Business Process Management (BPM) [1–3] and Systems (BPMS) [4], with Service Oriented Computing (SOC) [5] and Architecture (SOA) [6, 7] proposals, has gained many adepts both in industry and academy, as an straightforward way to connect Business Processes (BPs) with the services implementing them. Nowadays collaborative and virtual organizations need increasingly support to enact their collaborative BPs, fulfilling both centralized and decentralized scenarios, which depend on the organizations involved and/or the restrictions of their collaborative BPs and technologies.

The SOC paradigm helps in solving this problem by designing specific software pieces i.e. services, providing desired qualities such as loose coupling, high cohesion, easing interoperability, normalizing data exchanges (i.e. types and format), explicitly defining interfaces for interactions between systems, among others. Also, collaborative BPs can be modeled and executed in a notation such as Business Process Model and Notation (BPMN 2.0) [8], or as services composition [6, 7] with the Web Services Business Process Execution Language (WS-BPEL) [9] if the execution is fully automatic (i.e. no user interactions).

Collaborative organizations sharing common goals [10] are focused on integrating their software systems in order to exchange data and execute business functions setting up their business processes (BPs) [1–3]. This collaboration can be of two main types: (i) organizations are part of a collaborative environment in which the interactions between BPs and services are explicitly defined and agreed; or (ii) organizations offer capabilities for integration, not explicitly agreeing on their BPs but mainly on the contract of the services they expose or require to be able to participate in the collaborative environment.

A BPMS provides a complete platform to execute BPs by means of human tasks interacting with users and automatic tasks invoking services when needed, allowing the interaction between different participants. Integration platforms are specialized middleware-based infrastructures, notably Enterprise Service Bus (ESB) to support the implementation of a SOA [11], providing an intermediate processing layer between applications and services with the goal of facilitating integration issues. This middleware can be integrated with BPMS platforms helping supporting both types of collaborative scenarios as defined in (i) and (ii).

Also, in order to support the realization of human tasks within the BPMS, it would be useful to include mainstream social platforms such as Facebook, Twitter, LinkedIn and Google+, for employees involved in the BPs and for external users interacting with the platform. These capabilities would help, for example, to notify users when some activities need to be performed by employees or when user's participation is required to complete a process activity, among others.

In this paper we present a proposal for setting up a collaborative environment, both methodological and technological, in which organizations (e.g. in e-government) can interact based on their BPs and services implementing them. At the center of the proposal we defined: (i) a reference architecture for a process-aware inter-organizational service integration platform (PA-IOSIP) which is mainly based on the conjunction of a BPMS platform with an integration middleware infrastructure (e.g. ESB) [12], and (ii) a maturity model which provides a roadmap to guide the efforts towards setting up such a collaborative environment between willing organizations, taking into account several dimensions such as the different type of participating organizations, their infrastructure, collaborative processes, among others.

The rest of the article is organized as follows: In Sect. 2 we present related work. In Sect. 3 we analyzed several collaborative scenarios to be supported by our PA-IOSIP. In Sect. 4 we describe our proposal including the definition of the maturity model, the dimensions and elements we have defined within them.

In Sect. 5 we present a case study based on the implementation of a collaborative BP carried out within the Uruguayan e-government, as a proof of concept. Finally in Sect. 6 we present some conclusions and future work.

2 Related Work

In [13] we presented a systematic literature review of existing approaches to support services and BPs lifecycle, with focus on modeling, design and execution, but few of them succeeded in linking services to the BPs lifecycle [14] in an integrated manner. From the technological point of view, some proposals support the execution of collaborative BPs and services, but mainly with their own implementations of BPs engines and platforms.

Several well-known maturity models were defined for software process improvement such as Business Process Maturity Model [15], Capability Maturity Model (CMM) [16] and the Capability Maturity Model Integration (CMMI) [17]. These are reference models for evaluating the maturity and capacity of an organization, with regards to key elements as defined in each case. We also analyzed several proposals for collaborative and integration frameworks and platforms, including specific proposals for e-government which constitutes our real context for prototyping the proposal [18–22].

In [19] an e-Government Maturity Model which integrates the assessment of technological, organizational, operational and human capital capabilities is proposed. The model is structured around four domains: “e-Government strategy”, “IT governance”, “process management” and “organization and people”. The “process management” domain comprises the following key domain areas: business process management, performance management, services to citizen and business, interoperability, compliance, and quality and security assurance.

In [20] a BPM maturity and adoption model is proposed to guide organizations to process maturity, identifying six-phases for BPM maturity and adoption: acknowledge operational inefficiencies, become process-aware, establish intraprocess automation and control, establish interprocess automation and control, establish enterprise valuation control and create an agile business structure.

Compared to our proposal, these models are broader so they are not as specific as ours in issues related to BPMS adoption and services interaction. Also, the application of these models is intended to be performed individually for each organization. In contrast, our proposal is intended to be applied to the whole collaborative environment, considering each participating organization.

3 Collaborative Environment Scenarios Analysis

The context of analysis is a group of organizations that collaborate through a Process-aware Inter-organizational Service Integration Platform (PA-IOSIP) as described in [12]. We analyzed several collaborative scenarios (which we defined based on our knowledge of the real context for e-government) with different

combinations of BPs maturity and infrastructure, and social media capabilities and challenges in the complexity of the context.

The integration platform defines three main layers [12]. The top layer corresponds to the User applications layer which provides users with several applications to interact with the BPs executing in the integration platform, and also with other components such as documents and business rules. The middle layer corresponds to the BPMS layer which provides components to support the BPs lifecycle including modeling, implementing, executing, evaluating and improving processes. Finally, the bottom layer corresponds to the Integration layer which includes components to facilitate different aspects (e.g. security, connectivity, metadata) for achieving an interoperable cross-organizational collaboration.

3.1 Scenarios for the Collaborative Environment

The analysis of scenarios for the collaborative environment was based on defining different combinations of the participant organizations capabilities and their interaction with the central integration platform. In Fig. 1 we show two of the scenarios analyzed: an initial scenario (with few capabilities) and a high level scenario (with more complex capabilities). The analysis starts with the initial scenario going up through the high level scenario by adding complexity and capabilities in the participant organizations, the integration platform and the infrastructure involved.

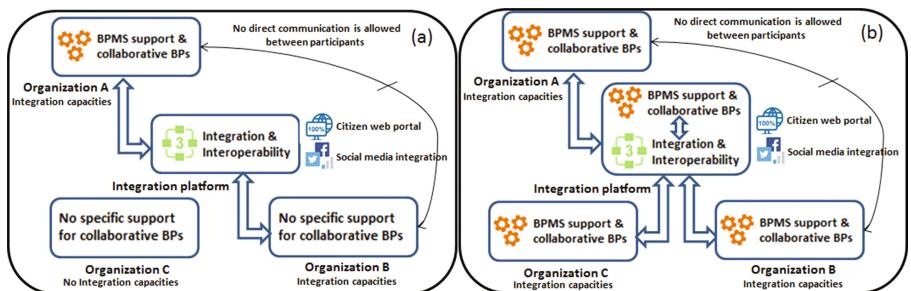


Fig. 1. Example of scenarios analyzed for the collaborative environment

As shown in Fig. 1(a), the first step to set up this collaborative environment is to provide an integration platform that allows participant organizations to interact within each other through it. This integration platform provides several components (e.g. security, connectivity, metadata) for achieving an interoperable cross-organizational collaboration, mainly based on services integration with no BPMS platform. There are some organizations that has integration capacities and therefore can be partially integrated in the collaborative BPs, and there are other organizations that does not present these capacities.

The most advanced organizations will present not only integration capacities but also provide BPMS support for collaborative BPs, interacting with other organizations by means of the integration platform. A web portal for interacting with clients/partners is mandatory even at this early stage, providing at least initiate and query capacities for their processes. Social media integration with users is desirable at least for external users from the web portal.

At the high level scenario shown in Fig. 1(b), the BPMS support for collaborative BPs will be spread within the complete collaborative environment (as a signal of BPM maturity), also adding a central BPMS at the integration platform. All participating organizations will present integration capacities, and BPMS support, although BPMS providers could not be the same for all participants. In [23] we propose a methodology that helps organizations in choosing the most adequate BPMS for their needs, based on their infrastructure and specific requirements (functional and non functional).

Processes execution will be hosted and carried out within the BPMS of the process owner in each case, interacting in a (mostly) asynchronous way with the rest of the BPMS via the central integration platform. The web portal for external users should allow specific queries regarding process execution, even if the execution is at that moment within another organization's control. For this, identifying the best way to chain the invocations through participants of the collaborative BP by means of the integration platform is a key element. Social media will be in place not only for external users, but also for organization's employees to execute tasks within the BPMS platform.

Between the initial scenario and the high level scenario several other intermediate scenarios take place, as the participant organizations continue growing their capacities to interact within the integration platform, to specify, model, implement and execute their own internal BPs in a BPMS platform and also taking part in the collaborative BPs defined at the integration level, including social media interaction with employees and external users, among other elements.

3.2 Scenarios for Social Media Interactions

As pointed out in [24] it is reasonable to combine BPM with social software as a way of improving users interactions. However, there are still few studies on how BPM benefit from social software in practice. As an example, in [25] the authors propose a BPMN extension for expressing social interactions like direct messaging, broadcasting and voting, but they did not take these ideas into action. In a collaborative environment, it is necessary to think about how external users (e.g. citizens in an e-government case) can interact with the execution of a BP from their social networks of daily use. In this context, we explored interaction mechanisms from three of the main social networks (Facebook, Twitter and Google+) within the execution phase of a BP. In Table 1 we identify common interactions between users and the execution of a BP through web media.

In the e-government context for example, it could be useful for a citizen to enter the web portal and view available processes, to query processes by name

Table 1. Common interactions through web access media

Category	Functionality	Interaction media
Processes	View available processes	Web site
	Query processes by name and category	Web site
	View process information	Web site
Cases/Instances	Create a process instance	
	List created instances for an user	
	View historical data from an instance	
Others	Login	
	Notifications	
	Documents	
	Confirmation	
	Voting	

and category with respect to its specific needs, and also view process information. From this list of processes, a citizen can create a process instance providing the information to initiate it, or use a specific social network for starting it, e.g. by sending a tweet or filling a Facebook or Google+ form. The citizen could also want to check the status of its request, thus it is important to list created instances for an user and provide a way to view historical data from an instance. These functionalities can be provided in a classical way from the E-Government portal, or for example from an embedded Facebook application which requires the user to use its network-specific credentials for login.

There are many kinds of notifications within processes. In this social environment, the user can follow the public profile of an organization to access news, or the process can send direct messages, e.g. a tweet to the user, or a Facebook message of a private publication in the wall. An interesting interaction arises when an organization needs to arrange a meeting with the user. In this case the user can receive an invitation to a private Google Calendar event, and also receive an automatic reminder when the meeting date is near. Alternative, a public calendar can be used.

Process information can take the form of documents which can both assessed when the user queries the information of a specific instance and from central repositories like Google Drive. In some cases, the user needs to send a confirmation with respect to some activity in which he/she is involved. This can be done by notifying the user by publishing a private message in its Facebook wall and then using the Like button. In the case of meetings, it is also possible to use Google Calendar confirmations. Finally, a social intensive functionality is **voting** in which users can take collaborative decisions or respond to surveys. The three social networks provide fully customizable forms for polling and processing their results.

4 Setting up a Collaborative Environment

Based on the analysis of collaborative environment scenarios we discussed in Sect. 3, and on the analysis of existing proposals, standards and approaches for the definition of integration platforms and collaborative environments, we have defined the following research question: *How can a collaborative environment between interacting organizations be set up considering both methodological and technological requirements by defining key elements that when achieved, guarantee an integration compliance level between participants?*

We have also defined sub-questions to guide our research work, mainly focused on the reference architecture for an integration platform, the maturity model and the case study for validation. We applied research methods suitable for each phase, starting with an extensive analysis of existing proposals (systematic literature review, existing standards and approaches for integration platforms, maturity models). For the definition of our proposal, we followed design science principles [26], creating artifacts: (a) reference architecture for an integration platform, (b) maturity model to guide the efforts in setting up a collaborative environment within (a), and (c) case study within the uruguayan e-government as a proof of concept for (a) and (b).

4.1 Reference Architecture for an Integration Platform

The reference architecture defines a Process-aware Inter-organizational Service Integration Platform (PA-IOSIP) [12], as introduced in Sect. 3. In particular, the platform provides support for collaborative BPs and services execution within a controlled distributed environment (e.g. e-government, e-health) where organizations communicate through a private network and there are formal agreements between participants. It can also be an open one (e.g. e-commerce, e-science) where participants collaborate via Internet without explicit agreements. The reference architecture for the integration platform details can be seen in [12].

4.2 Maturity Model for Collaboration

The maturity model we have defined is based on key definitions of the maturity models BPMM, which in turn is based on the CMM and CMMI, taking into account the analysis of scenarios in Sect. 3. The main objective of the maturity model is to guide the efforts to set up a collaborative environment within the reference architecture for the integration platform, by evaluating the maturity of the whole set from the point of view of the integration platform. For this, we considered meaningful dimensions that we have identified from the scenarios analysis. The maturity model defines five levels of maturity, following the definitions of the maturity models of reference. In Table 2 we present the general definition of the maturity model.

We identified five dimensions of interest for the analysis: (i) Maturity of each participant organization, (ii) Level of integration and interoperability within the platform, (iii) Support for BPMS and collaborative BPs, (iv) Central portal

Table 2. Levels defined in the maturity model for a collaborative environment

Level	Description
5 - Optimized	the measures defined, collected and analyzed in level four are used for continuous improvement of the collaborative BPs execution, services execution or infrastructure support
4 - Predictable	adds the definition, collection and analysis of measures for level three elements. These measures are both functional (e.g. BPs tasks and BP cases execution) and non functional (e.g. BPs and services execution: response time, throughput and security)
3 - Collaborative	the collaborative environment is established, with a central BPMS and middleware to manage interactions with participant organizations. Each participant includes a BPMS platform for their internal processes and parts of the collaborative ones
2 - Integrated	participant organizations are mainly connected by means of P2P interactions within each other, and/or by means of a central interoperability middleware supporting and managing interactions
1 - Initial	there is no integration between organizations, tasks are carried out based on forms, documents and/or paper. BPs are implicit in systems and/or people's knowledge

for external users interaction and (v) Support for social media integration. The summary of the maturity model is presented in Fig. 2 showing the levels and dimensions.

Maturity of Each Participant Organization. It defines the capacity that each organization has to interact with the integration platform and collaborate with other participants. The maturity of a participant organization is based on BPMM definitions and the extensions we made: (i) infrastructure support for the integration and interoperability within the platform, and (ii) support for collaborative BPs execution in a BPMS platform. The compliance for each level is measured by the percentage of the organization BPs that are supported.

Level of Integration and Interoperability Within the Platform. This dimension looks at the collaborative environment from the integration platform point of view. Bearing in mind the reference architecture we have defined for such platform, including BPMS and middleware, we define it at each maturity level considering the number of participant organizations that are integrated and effectively collaborating through the integration platform.

Support for BPMS and Collaborative Process. This dimension also takes the integration platform point of view. We evaluate the support for collaborative BPs provided by the general collaboration by means of BPMS support for BPs

	 Organizations Maturity	 Integration and Interoperability	 Support for BPMS and collaborative processes	 Citizens Global Portal	 Social Integration
LEVEL 1 INITIAL	Incomplete Without Restrictions	Incomplete Without Restrictions	Incomplete Without Restrictions	Incomplete Without Restrictions	Incomplete Without Restrictions
LEVEL 2 INTEGRATED	75% of Organizations in Level 2 or greater	Performed 25% of Organizations integrated through the integration platform	Performed Central BPMS hosts 25% of collaborative processes	Performed 25% of collaborative processes online	Performed Social component not integrated to the platform
LEVEL 3 COLLABORATIVE	75% of Organizations in Level 3 or greater	Managed 50% of Organizations integrated through the integration platform	Managed Central BPMS hosts 50% of collaborative processes	Managed 50% of collaborative processes online	Managed Social component Integrated: publications
LEVEL 4 PREDICTABLE	75% of Organizations in Level 4 25% of the rest in Level 3 or greater	Defined 100% of Organizations integrated through the integration platform	Defined Central BPMS hosts 75% of collaborative processes	Defined 100% of collaborative processes online	Defined Social component integrated: publications, notifications, authentic.
LEVEL 5 OPTIMIZED	75% of Organizations in Level 5 25% of the rest in Level 4 or greater	Defined 100% of Organizations integrated through the integration platform	Defined Central BPMS hosts 100% of collaborative processes	Defined 100% of collaborative processes online	Defined Social component integrated: publications, notifications, authentic.

Fig. 2. Maturity model with dimensions and definitions for each level

execution within all organizations and the integration platform. The final target is to provide a central BPMS within the integration platform to host general collaborative BPs, and also BPMS support in each participant organization, decentralizing all processes.

Central Portal for External Users Interaction. This dimension takes into consideration the support provided by the collaborative environment to external users (e.g. citizens) to interact with it. A central portal allows to initiate BP cases, query existing cases to retrieve the current state, the current executing task, among others. To do so, a key element is to define a traceability mechanism for collaborative BPs execution, for example, where each BP registers key data when it gives control to another participant.

Support for Social Media Integration. This dimension takes into account the support the integration platform provides for interaction via social media mainly for external users and organization's employees. The BPMS platform must be able to include for example notifications via twitter and messages and publications via facebook, providing users with a friendly way to track and be informed about the execution of their BPs, also interacting via the web portal with these platforms.

5 Proof of Concept

We have implemented a prototype of a real collaborative BP within the Uruguayan e-government [27], the “Born Alive” process, to develop a proof of concept of the reference architecture applying the maturity model we have defined. For doing so, we have integrated different technologies simulating the

context of the integration platform and participant organizations. The collaborative BP involves several government agencies such as: Health Ministry (owner), Social Security Institute, Social Services Ministry, National Registry. We have already analyzed this collaborative BP but from another point of view in [28].

We will focus here in the description of the infrastructure distribution, integration and capabilities provided by the participant organizations and the central integration platform, by means of implementing the reference architecture at the maturity model level Three. Further validation of the maturity model will include the definition and analysis of measures to be able to navigate to levels Four and Five for continuous improvement of collaborative BPs, infrastructure and services. We implemented the reference Architecture with two open source BPMS: Activiti BPMS¹ for the Health Ministry, and Bonita BPM² for the central integration platform, and as middleware platform the Red Hat Jboss Fuse³. We selected Activiti and Bonita since they provide similar functionalities with a very different approach (Activiti with focus on java developers and Bonita with focus on automating the development), to evaluate their integration in a real collaborative environment. In Fig. 3 we present the general definition for the prototype implementation.

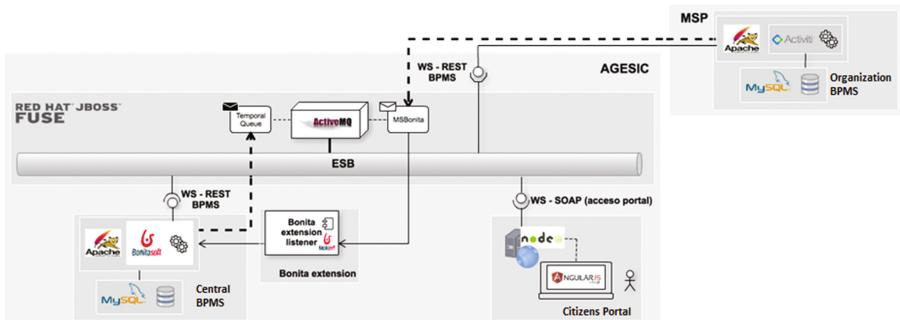


Fig. 3. Architecture and technologies used for the prototype implementation

Both Bonita and Activiti BPMS provide a REST API which allows interaction with the process engine from the middleware platform and the web portal. The integration platform exposes services as SOAP Web Services (mainly for security reasons) and invokes the REST API from the central BPMS (Bonita) or the MSP one (Activiti) when needed. All interactions between participants (with each other or the integration platform) goes through the middleware platform (as defined in the e-government infrastructure). Invocations and answers to/from other participants are simulated to execute the process completely. The

¹ Activiti BPMS Platform. <https://www.activiti.org/>.

² Bonita BPM. <http://www.bonitasoft.com/>.

³ RH Jboss Fuse. <https://www.redhat.com/en/technologies/jboss-middleware/fuse>.

middleware platform has definitions for routing the messages received within the SOAP WS, to the corresponding message queue and/or to invoke the REST API of the corresponding BPMS (e.g. to send a message).

We have also prototyped the social interactions described before with respect to the BP executed in Activiti BPMS. Social networks interact with the process engine through the REST API provided by Activiti, and every interaction from the process is encapsulated in Java classes in such a way that internal users of the e-gov platform are unaware that they are interacting with social networks. We have created specific social network accounts representing governmental dependencies and we configured them so they can use external applications that we provided for some specific interactions. In reply, the social networks provided us with authentication tokens that Activiti needs to interact through the WS they provide for interaction.

The proof of concept allowed us to put in place the needed infrastructure to set up a collaborative environment of maturity level three, following the definitions and guides defined in the maturity model. The middleware platform was integrated seamlessly with Bonita as the central integration platform, and Activiti was deployed as part of one organization, interacting with the integration platform, via SOAP/REST WS and messages. We tested integration with social networks for both external users and employees. We conclude that the definitions in the maturity model help defining what is needed to set up a collaborative environment.

6 Conclusions

This paper presented a proposal for setting up a collaborative environment to support the interaction of organizations within an integration platform. We defined a reference architecture for such an integration platform, and a Maturity Model which provides a roadmap for implementing collaborations within it. The maturity model provides key guide for the progressive implementation of the reference architecture in a collaborative environment, and how to evolve it to improve both infrastructure and collaborative BPs support.

Several scenarios were analyzed to help identifying the main elements to be considered by the model. The proposal was applied for the implementation of a collaborative BP within the Uruguayan e-government, which served as a proof of concept for the proposal, including social media integration. We believe that the proposal constitutes a step forward in providing guidelines to set up a collaborative environment. Future work will be extending the case study to higher maturity levels, and to carry out more case studies in other real environments, to continue evaluating the usefulness of the proposal.

Acknowledgement. We would like to thank students Verónica Gamarra and Rodrigo Lavista.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business process management: a survey. In: van der Aalst, W.M.P., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44895-0_1
2. Weske, M.: Business Process Management. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-642-28616-2>
3. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-33143-5>
4. Chang, J.F.: Business Process Management Systems: Strategy and Implementation. Auerbach Publications, Taylor & Francis Group, Boca Raton, New York (2005)
5. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. *Int. J. Coop. Inf. Systems.* **17**(2), 223–255 (2008)
6. Erl, T.: Service-Oriented Architecture: Analysis and Design for Services and Microservices, 2nd edn. Prentice Hall, Boston (2016)
7. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service Oriented Architecture Best Practices. Pearson Education, Upper Saddle River (2005)
8. Object Management Group (OMG): Business Process Model and notation (BPMN) v2.0 (2013). <http://www.omg.org/spec/BPMN>
9. WS-BPEL, OASIS (2008). <http://docs.oasis-open.org/wsbpel/2.0/>
10. Mylopoulos, J.: Cooperative Inf. Systems. *IEEE Expert.* **12**(5), 28–31 (1997)
11. Papazoglou, M.: Web Services and SOA: Principles and Technology, 2nd edn. Pearson Education Canada, Toronto (2012)
12. Delgado, A. and González, L., Ruggia, R.: A process-aware inter-organizational service integration platform to support collaborative organizations. In: IEEE International Conference on Services Computing (SCC), pp. 844–847 (2016)
13. Delgado, A., Ruiz, F., de Guzmán, I.G.-R., Piattini, M.: Main principles on the integration of SOC and MDD paradigms to business processes: a systematic review. In: Cordeiro, J., Virvou, M., Shishkov, B. (eds.) ICSOFT 2010. CCIS, vol. 170, pp. 88–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-29578-2_6
14. Delgado, A.: A services lifecycle to support the BPs lifecycle: from modeling to execution and beyond. In: IEEE International Conference on Services Computing (SCC), pp. 831–835 (2016)
15. Object Management Group (OMG): Business Process Maturity Model (BPMM) (2008). <http://www.omg.org/spec/BPMM>
16. Software Engineering Institute (SEI): Capability Maturity Model (CMM) (1993). <https://www.sei.cmu.edu/reports/93tr024.pdf>
17. Software Engineering Institute (SEI): Capability Maturity Model Integration (CMMI) (2006). <http://www.sei.cmu.edu/downloads/cmmi/10tr033.docx>
18. Mecella, M., Batini, C.: Enabling Italian e-government through a cooperative architecture. *Comput. J.* **34**(2), 40–45 (2001)
19. Valdés, G., Solar, M., Astudillo, H., Iribarren, M., Concha, G., Visconti, M.: Conception, development and implementation of an e-Government maturity model in public agencies. *J. Gov. Inf. Q.* **28**(2), 176–187 (2011)
20. Kerremans, M.: Maturity assessment for business process improvement leaders: six phases for successful BPM adoption, Gartner, Stanford, pp. 7–15 (2008)

21. Bhat, J., Fernandez, J.: A holistic adoption framework for long term success of BPM, BPM and workflow handbook digital edition v2 (2008)
22. Rosemann, M., De Bruin, T., Hueffner, T.: A model for business process management maturity. In: ACIS 2004 Proceedings (2004)
23. Delgado, A., Calegari, D., Milanese, P., Falcon, R., García, E.: A systematic approach for evaluating BPM systems: case studies on open source and proprietary tools. In: Damiani, E., Frati, F., Riehle, D., Wasserman, A.I. (eds.) OSS 2015. IAICT, vol. 451, pp. 81–90. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17837-0_8
24. Erol, S., Granitzer, M., Happ, S., Jantunen, S., Jennings, B., Johannesson, P., Koschmider, A., Nurcan, S., Rossi, D., Schmidt, R.: Combining BPM and social software: contradiction or chance? *J. SW Maint.* **22**(6–7), 449–476 (2010)
25. Brambilla, M., Fraternali, P., Vaca, C.: Combining social web and BPM for improving enterprise performances: the BPM4People approach to social BPM. In: 21st World Wide Web Conference, pp. 223–226 (2012)
26. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Q.* **28**(1), 75–105 (2004)
27. González, L., Ruggia, R., Abin, J., Llambías, G., Sosa, R., Rienzi, B., Bello, D., Álvarez, F.: A service-oriented integration platform to support a joined-up e-Government approach: the Uruguayan experience. In: Advancing Democracy, Governement and Governance, pp. 140–154 (2012)
28. Ruggia, R., Delgado, A., Abin, J., González, L., Garbusi, P.: Managing consistency in e-government transactions: the case of Uruguay. In: 9th International Conference on Theory and Practice of Electronic Governance (ICEGOV), pp. 313–322 (2016)



Toward an Interactive Mobility Assistant for Multi-modal Transport in Smart Cities

Christian Kuster¹(✉), Nils Masuch², and Fikret Sivrikaya¹

¹ GT-ARC gemeinnützige GmbH, Berlin, Germany
christian.kuster@gt-arc.com

² DAI-Labor, Technische Universität Berlin, Berlin, Germany

Abstract. The worldwide trend of urbanization necessitates new forms of travel within and between megacities. Shared mobility and multi-modal mobility approaches have been on the rise to address this need, but existing solutions are often limited by proprietary systems that cannot be combined automatically. On the other hand, the concepts of *Smart Cities* and the *Internet of Things* offer great opportunities to interconnect these different services, whereas such holistic approaches lead to new challenges for emerging applications. In this paper we highlight the challenges and provide a roadmap towards an agent-based interactive mobility assistant for multi-modal transport in smart cities that grounds on the *Belief-Desire-Intention (BDI)* model, an approach for deliberative agents using mental attitudes, in order to overcome the information overload and proactively help travelers along their way.

1 Introduction

The ever increasing urbanization has significantly aggravated the traffic situation in megacities. Currently around 75% of all citizens in the European Union are living in urban areas. New mobility solution approaches emerge to overcome these issues and to adapt to the new habits of especially the younger population. One example is the growing area of *shared mobility*, which is commonly offered in the form of car-, bike-, or ride-sharing services. While the increased variety of mobility options improves the efficiency of transportation in large cities, it also complicates the residents' mobility planning. To address this complexity, *multi-modal mobility solutions* that integrate different travel modalities have started emerging on the market. Nevertheless, these approaches again employ isolated applications. Instead of alleviating the complexity in travel planning, they even aggravate it with new forms of unfamiliar services and information.

In an even wider context, the concept of a *Smart City* and the heavily linked topic of *Internet of Things (IoT)* [16] bring even more possibilities to a future travel system. Sensors can, for example, predict weather and congestions or create health maps for biking/walking. This means that a useful application

should not only regard mobility service providers, but have the ability to utilize other information sources in novel and perhaps unusual ways.

The recently completed *Intermodal Mobility Assistance (IMA)* project [9] has aimed at providing an open and scalable platform that enables users to seamlessly integrate multiple service providers in their route planning. Service providers, on the other hand, can easily join the platform to provide mobility or information services. However, routes suggested by an application using this platform and all other available travel-related data are too much information for the traveler to have an overview. Services may provide basic information or operations that are individually neither useful nor comprehensible to a human, but may become very useful when used in an aggregated way. Hence, especially on such open platforms that combine various services, an assisting, consolidating entity is needed to keep the system manageable for users and to generate an added value. This calls for an automated and proactive personalized mobility assistant that helps before and during travel. To address this need, we propose a BDI-style [7] agent-based solution, the *Interactive Routing Assistant (IRA)*.

A first version of IRA has been realized in the context of IMA project, with the goal of monitoring individual users traveling over inter-modal routes provided by the IMA platform. Implemented using JIAC V [12], many trade-offs had to be made due to the novelty and complexity of the platform that had to be realized within the project's life span. While being coupled loosely to the BDI model, the components of the IRA agent as well as the platform's domain ontology and service interfaces were predefined, fixed, and implemented with Java concepts.

With the experience gained from IMA, we argue that a more systematic and elaborate approach is needed, in which an independent assistant collects information from heterogeneous sources and chooses between different services to fulfill its goals, in order to face the challenges behind it, as discussed in Sect. 3. Our approach for the agent-based IRA is then provided in Sect. 4.

2 Related Work

While the number of approaches and products that deal with today's problems in the personal transportation in urban areas and especially multi-modality is rising, most of these focus on calculating routes [10] or traffic control & congestion management [4]. The need for a mobile companion is often neglected.

Analyzing related research articles about virtual assistants in the transport domain, we realize that most approaches focus on humans with special needs [1] or implement simplistic [14] or particular [15] use cases; the universal need for such assistants as we envision and as studies show [2] is not well addressed so far. An approach that targets a wider spectrum of users and contexts is shown in [6], where public transport infrastructures such as vehicles and stations are enriched with service units that can communicate with a mobile application to give the user additional contextual information. The presented approach again relies on a closed system architecture and provides only generic and reactive services.

The interactive mobility assistant proposed in this work focuses on the connection of BDI model with Smart Cities in the form of its constituent web

services. A good introduction is given in [5] discussing what distinguishes agents from web services and what problems may arise when enabling BDI agents to utilize web services.

An approach that is similar to our concept is presented in [11]: An ontology-based BDI agent is introduced that is capable of using semantic web services in heterogeneous environments to create so-called workflows to fulfill its goals. The agent includes two types of ontologies, namely the operational ontologies that are domain-independent and responsible for the agent to be functional, and the application ontologies describing the domain. One of the application ontologies is the so-called *domain workflow pattern ontology* that contains predefined workflows for specified tasks within the domain. However, the authors do not address how the service discovery and matching is done; furthermore, a static web with well-defined workflows is assumed.

3 Research Challenges

Implementing an intelligent assistant that helps travelers to overcome the tremendous amount of information brings up different but entangled challenges to be solved, which we want to address in this section, gathered by the study of the aforementioned relevant literature as well as our own experience from the past projects.

- (1) *Understanding of (web) services*: To build an intelligent system that is capable of automatically understanding the meaning of the exchanged information and using services, complex ontology representations and rules that allow detailed reasoning are needed.
- (2) *Aggregation of services*: To create novel value-added utilizations of available information from services according to the own domain and intentions, the intelligent system needs a computation model that is able to break down goals into subsets that can be mapped onto the found services.
- (3) *Highly dynamic environment*: Smart Cities are highly dynamic; services join and leave, meaning that plans made by aggregated services can fail eventually or new and better alternatives become available. Even new types of information can lead to new assumptions and rules for the system.
- (4) *Heterogeneous environment*: Contrary to past distributed systems, very diverse set of devices and services, along with their own data models and interface protocols, can be found in Smart Cities and IoT. Therefore, intelligent systems must have the ability to abstract from these and unify the knowledge [8].
- (5) *Conflicting beliefs*: Since the system utilizes different information sources, it can happen that conflicting beliefs occur, e.g., the position of the traveler indicates the usage of a bus near a railway track, while the mobile device sensors indicate the usage of a train. Strategies to resolve such conflicts are needed.

4 Research Approach

To tackle the challenges mentioned above we propose an ontology-based BDI agent architecture that uses semantic web services to retrieve relevant information and perform actions. The approach both addresses the agent model as well as the agent's environment.

4.1 The Agent Model

The agent is based on the well-known BDI model and uses the core components of *Belief*, *Desire* and *Intention* to model its internal state. In addition, it has sensors and actuators to perceive and influence its environment. In our approach the beliefs are retrieved from external services as well as from the mobile device the assistant is executed on. The agent's intentions (selected plans from the plan database) are also located in the external services and the mobile device. The details of the external environment are discussed in Sect. 4.2. The goals (desires) are encoded in the saved travel routes: IRA wants to ensure that the user arrives at their destination in time. As travel routes can be divided into sub-routes, the corresponding goal can be divided into sub goals, leading to a goal hierarchy. To enable the agent to process new types of information and aggregate them to form new knowledge (Sect. 3–(2)), the BDI model within the agent is implemented using ontologies. To enable the agent to handle semantic web services it needs a corresponding ontology incorporated into the BDI ontology and links both beliefs and intentions with their web service representation. The belief base is monitored using observers; changes in the beliefs trigger rules that either lead to new beliefs or the selection or discarding of intentions. As already mentioned in Sect. 3–(5) the belief base can store beliefs that contradict each other, so they are extended by the notion of certainty; the value can be determined regarding the belief-delivering service and by internal deduction rules (e.g., stable knowledge becomes more certain over time). Lastly, since a travel route is heavily influenced by the user's preferences and habits, a user ontology is used to model beliefs about the user, which can then weight and modify selected plans. An overview of the agent can be seen in Fig. 1.

4.2 The Agent Environment

The agent uses external semantic web services to perceive information about its environment and to perform chosen plans. Furthermore, the user's mobile device that runs the mobility assistant is also employed for both perceiving and interacting (see Fig. 1). The route planner is an external service that can be requested for specific routes. In our approach, coming from the IMA platform, the route planner is based on a multi-modal approach, in which different service providers can be included in a request; this way IRA can focus on the on-route management of the traveler. Our approach of course allows multiple route planners to be accessible; IRA would then choose the most appropriate one according to its plans. Alongside with the route planner different types of services are present

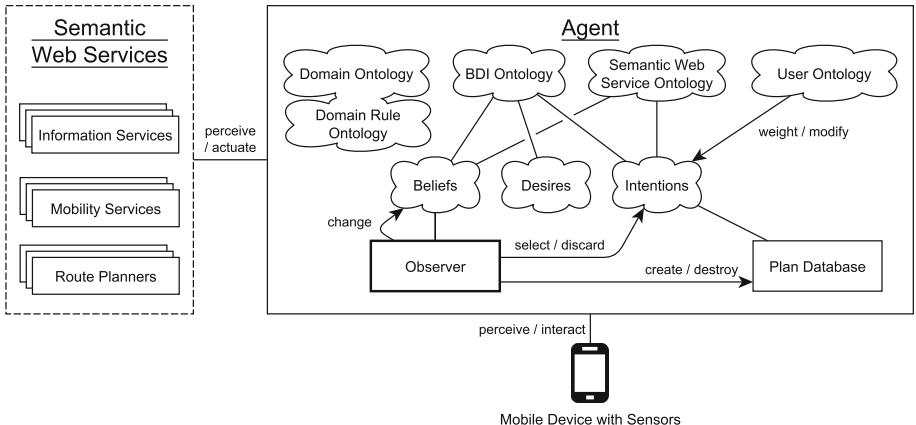


Fig. 1. Structure of the agent

in the agent's environment. They can be divided into *transportation-providing services*, like public transport, car-sharing, bike-sharing, etc., and *information-providing services* that do not necessarily have to be located in the transportation domain, such as parking space availability, traffic congestion, weather prediction, air pollution, tourist features, etc.

All these services have a semantic description based on OWLS-S [3] that can be parsed by the agent (Sect. 3–(1))[13]. In this work we assume the ontology to be fixed to the transport domain, therefore we do not have to handle the possible ontology merging problem (Sect. 3–(4)). Still, we have to make some assumptions about the service description format: Since we model an assistant that is responsible for a real-world traveler with changing world states, services should provide information regarding how to react to failed plans, e.g., compensation actions (In a car-sharing scenario: cancel the booked car).

5 Conclusion and Future Work

With the adoption of smart city technologies, there is strong potential for proactive smart entities for urban mobility systems, such as our proposed multi-modal mobility assistant. While the benefits of such applications are already recognized by the research community, there are still many issues to be resolved. We have argued that our approach is able to face the challenges of future developments. The next steps are to further work out the developed concepts and implement them into our existing framework. We plan to further generalize the service request mechanism and develop the proposed ontologies for a more abstract agent that complies even better with the BDI approach.

Acknowledgment. This work is supported in part by the *German Federal Ministry of Education and Research (BMBF)* under the funding reference numbers 01IS12049 & 16KIS0580.

References

1. Barbeau, S., Labrador, J., Winters, P., Perez, H., Georggi, N.: The travel assistant device: utilizing GPS-enabled mobile phones to aid transit rides with special needs. In: Proceedings of the 15th World Congress on Intelligent Transportation System, NY (2008)
2. Beul-Leusmann, S., Samsel, C., Wiederhold, M., Krempels, K.-H., Jakobs, E.-M., Zieffle, M.: Usability evaluation of mobile passenger information systems. In: Marcus, A. (ed.) DUXU 2014. LNCS, vol. 8517, pp. 217–228. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07668-3_22
3. Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: semantic markup for web services, November 2004. <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
4. Chen, B., Cheng, H.H.: A review of the applications of agent technology in traffic and transportation systems. IEEE Trans. Intell. Transp. Syst. **11**(2), 485–497 (2010)
5. Dickinson, I., Wooldridge, M.: Agents are not (just) web services: considering BDI agents and web services. In: Proceedings of the SOCABE 2005 (2005)
6. García, C.R., Candela, S., Ginory, J., Quesada-Arencibia, A., Alayón, F.: On route travel assistant for public transport based on android technology. In: 2012 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 840–845. IEEE (2012)
7. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In: Müller, J.P., Rao, A.S., Singh, M.P. (eds.) ATAL 1998. LNCS, vol. 1555, pp. 1–10. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49057-4_1
8. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. Knowl. Eng. Rev. **18**(1), 1–31 (2003)
9. Keiser, J., Masuch, N., Ltzenberger, M., Grunewald, D., Kern, M., Trollmann, F., Acar, E., Ç. A. Salma, Dang, X.T., Kuster, C., Albayrak, S.: IMA - an adaptable and dynamic service platform for intermodal mobility assistance. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1521–1528, October 2014
10. Li, J.Q., Zhou, K., Zhang, L., Zhang, W.B., et al.: A multimodal trip planning system incorporating the park-and-ride mode and real-time traffic/transit information. In: Proceedings of the ITS World Congress, vol. 25, pp. 65–76 (2010)
11. Liu, C.H., Chen, J.J.Y.: Using ontology-based BDI agent to dynamically customize workflow and bind semantic web service. JSW **7**(4), 884–894 (2012)
12. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., Keiser, J., Burkhardt, M., Kaiser, S., Albayrak, S.: JIAC V: a MAS framework for industrial applications. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, pp. 1189–1190. International Foundation for Autonomous Agents and Multiagent Systems (2013)
13. Masuch, N., Kuster, C., Albayrak, S.: Semantic service manager-enabling semantic web technologies in multi-agent systems. In: Proceedings of the Joint Workshops on Semantic Web and Big Data Technologies, INFORMATIK 2014, Stuttgart, Germany, pp. 499–510 (2014)

14. Papangelis, K., Sripada, S., Corsar, D., Velaga, N., Edwards, P., Nelson, J.D.: Developing a real time passenger information system for rural areas. In: Yamamoto, S. (ed.) HIMI 2013. LNCS, vol. 8017, pp. 153–162. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39215-3_19
15. Rehrl, K., Bruntsch, S., Mentz, H.J.: Assisting multimodal travelers: design and prototypical implementation of a personal travel companion. *IEEE Trans. Intell. Transp. Syst.* **8**(1), 31–42 (2007)
16. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. *IEEE Internet Things J.* **1**(1), 22–32 (2014)

PhD Symposium

PhD Symposium Preface

Service-oriented computing (SOC) has rapidly evolved with technologies such as Web services, Cloud services, and the Internet of Things. While this has provided the industry and practitioners with the opportunities for a new generation of products and services, it has also raised many fundamental research challenges and open issues. The International Conference on Service Oriented Computing (ICSOC) is the premier international forum for academics, industry researchers, developers, and practitioners to report and share groundbreaking work in service-oriented computing.

The International PhD Symposium on Service Computing was in conjunction with the 15th International Conference on Service Oriented Computing (ICSOC 2017) on November 13–16, 2017 in Málaga, Spain.

The ICSOC PhD Symposium is an international forum for PhD students working in all the areas related to the service computing. Its goals are:

- To bring together PhD students and established researchers in the field of service oriented computing,
- To enable PhD students to interact with other PhD students and to stimulate exchange of ideas, suggestions, and experiences among participants,
- To provide PhD students an opportunity to present, share and discuss their research in a constructive and critical atmosphere, and
- To provide PhD students with critical and constructive feedback from experts on their already completed and, more importantly, planned research work.

To achieve these goals, the symposium operates in a workshop format, giving PhD students an opportunity to showcase their research and providing them with ample feedback from senior international researchers and peer PhD students. Apart from the presentation sessions, this year, the program of the symposium also incorporated a keynote and a panel. The keynote entitled *Demystifying Smart Data & Smart Industrial-Purpose Applications: Solving Problems & Creating Opportunities* was given by Prof. Michael Papazoglou. The panel was composed by three experts and a moderator, in which the panelists, guided by the moderator, discussed topics related to the PhD development and its employment opportunities.

The symposium in Málaga, Spain is the 13th edition of the series held in conjunction with the ICSOC conferences in Banff (2016), Goa (2015), Paris (2014), Berlin (2013), Shanghai (2012), Paphos (2011), San Francisco (2010), Stockholm (2009), Sydney (2008), Vienna (2007), Chicago (2006), and Amsterdam (2005).

Each submission was reviewed by three members of the program committee and after a thorough review process, 7 papers out of 9 were accepted to constitute the program of the PhD symposium.

We gratefully acknowledge the support of the contributors to this PhD symposium and express our great esteem to the program committee members for the time and effort they have put in reviewing papers.

Organization

Keynote

Michael Papazoglou Tilburg University, The Netherlands

Panelists

Juan Manuel Murillo	University of Extremadura, Spain
Juan M. Vara	King Juan Carlos University, Spain
Flavio de Paoli	Universit di Milano-Bicocca, Italy

Program Committee

Pedro lvarez	Universidad de Zaragoza, Spain
Francis Palma	Concordia University, Montreal, Canada
Hoa Dam	University of Wollongong, Australia
Massimo Mecella	University of Rome, Italy
Gustavo Rossi	UNLP, Argentina
George Spanoudakis	City University London, UK
Mathias Weske	University of Postdam, Germany
Xiwei Xu	CSIRO, Australia
Pascal Poizat	Universit Paris Ouest, France



Meeting IoT Users' Preferences by Emerging Behavior at Run-Time

Daniel Flores-Martin^(✉)

University of Extremadura, Cáceres, Spain
dfloresm@unex.es

Abstract. Internet of Things systems are increasing their importance in our lives. To provide their maximum benefit, they must be manually configured according to the users' needs and routines. Thus, the increasing number of smart devices and systems being deployed will make this task completely unmanageable in the near future. This could limit the rise and penetration of IoT. Moreover, smartphones are standing out as the interface through which people interact with these systems. Due to their increasing capabilities they can also detect and analyze their users' daily activities. Therefore, this research tries to address this situation by proposing an architecture that allows smartphones to learn from the habits of their users through automatic learning techniques, and a programming model that allows run-time adaptation of the IoT systems behavior to the detected needs through the invocation of the services provided by the smartphones.

Keywords: Internet of Things · Context · Smartphones
Machine learning

1 Introduction

The relevance of the Internet of Things (IoT) increases as more and more connected devices are developed. One of the general purposes of these devices is to make people's life easier, simplifying tasks or executing them automatically.

For these systems to achieve their goals, they must fulfill their users' needs. For that, they have to be manually configured by the users, who often do not have the necessary technical skills, with the consequent investment of time and frustration this can cause. This, which today may be a serious drawback, in the near future in which it is estimated to be 50–100 billion of smart devices [10], may be unmanageable. Likewise, it is not enough to configure the devices once at the beginning, but also users must reconfigure them as their habits and needs change.

Supervised by: Javier Berrocal (jberolm@unex.es), Jose Garcia-Alonso (jgaralo@unex.es) and Juan M. Murillo (juanmamu@unex.es). University of Extremadura, Cáceres, Spain.

These drawbacks may be addressed by developing software capable of adapting its behavior to the people's needs.

Similar problems have already been detected by other researchers [5, 13]. Additionally, several research areas can contribute to solve them, specifically Context-Oriented Programming (COP), Ambient Intelligent (AmI) and Machine Learning (ML).

Therefore, the problem statement faced in this PhD. work is that the interactions between people and devices requires too much attention and time from people. With the growth predictions of IoT in the near future, this demand may become unmanageable.

The alignment of this proposal with ICSOC corresponds to the development of technology that allows the creation of services to be deployed in smartphones, attending to the people's preferences and the environment that surrounds them.

The rest of the paper is structured as follows. Section 2 describes related fields and hypothesis about our proposal. Section 3 presents the main objectives of this research. Next, Sect. 4 shows the preliminary results. Finally, Sect. 5 details the evaluation plan of this research work.

2 Related Fields and Hypothesis

This work main hypothesis is the following: By using paradigms such as AmI, COP and ML; and smartphones, that learn about their owners habits and preferences, the interactions between users and IoT systems can be automatized according to the users' preferences.

To address this, smartphones, in combination with the predictive models created through ML techniques, could be used to discover patterns of behavior. This can be done by using one of the most popular ML branches, Deep Learning (DL) [8], taking as input the data gathered by the smartphones about their owners' activities.

By analyzing the information gathered by the smartphones' sensors, we can identify what actions people are doing, at what time, where or when, and which IoT devices they interact with. In this way, we could be able to automate tasks, or predict future behaviors.

Several research works support this hypothesis. First, AmI tries to make everyday environments sensitive and adaptable to people [12]. However, AmI needs to know the users' preferences to establish when a device should act. This is a complex problem when the needs of several people have to be analyzed, specially when their presence was not originally planned within the system.

Second, COP [6] allows software developers to define the behavior of the applications, allowing to activate or deactivate certain behaviors or functionalities depending on the contextual information. However, these programming models require the different behavior to be defined at design time [7]. Behavior depend on users' needs and preferences, as well as changes over time. For this reason, it can be difficult and inefficient, to try to anticipate all possible behaviors at design time. Given this, the possibilities of adaptation of the software would be limited to a certain number of predefined contexts in the source code.

Third, the supervisors of this PhD. work are focusing on solutions to improve the integration between people and IoT systems through the use of smartphones, such as People as a Service (PeasS) [4] and Internet of People (IoP) [9].

Finally, attending aspects of human activity, Cook et al. [11], proposed an approach for recognizing complex activities of daily living, by using body-worn sensors and ambient sensors, to provide additional hidden context.

We are aware that there are many proposals for software development whose behavior is adapted to the context, but they do not cover in many cases the above mentioned problems. Thus, the research challenges that we address are several. First, the lack of a unified model of person-IoT interaction. IoT devices are produced by various manufacturers, each with their own interaction model. Second, the lack of an automatic negotiation model for the interaction between people and IoT devices according to the people's preferences. Third, the absence of learning models of people's preferences from their interactions with IoT devices. If these problems could be solved, a better integration of people in IoT environments would be obtained.

3 Research Objectives

In addition to the general objective that follows directly from the hypothesis above, this is to achieve a better integration of people with the IoT by making the devices learn about their owners. This thesis has the following specific objectives:

- (1) **Design an architecture where people and connected devices are represented.** The architecture must support the identification of the people and devices that are present in a situation, their connections and the automatic configuration of the devices' behavior according to the preferences of the people. The main contribution to be derived from this objective will be an infrastructure that can contribute to the definition of a standard for connecting people with devices independently of the manufacturer.
- (2) **Propose a learning model about people's preferences.** This model will be hosted on smartphones and must, therefore, meet the resource restrictions of these devices. It will be provided as a service from the smartphones to the architecture in order to make the preferences of all the involved people known. The main scientific contribution associated with this objective will be ML algorithms adapted to be executed in smartphones and prepared to manage an important and larger volume of data coming from a single person. By labeling these data at the beginning, we will study people's routines, and use them to train DL models to predict people's future routines.
- (3) **Propose a programming model for the architecture.** Based on the capabilities of each device, it is possible to develop software that will satisfy the people's preferences. The main contribution associated with this objective will be a programming model that, besides allowing programming strategies for specific contexts, will allow strategies to emerge once a context occurs. Neither the context or the strategy will be defined at design time.

The next section shows our architecture, detailing how through the context and the user profiles, it could facilitate the handling of IoT devices.

4 Preliminary Results

This work has already achieved some preliminary results regarding the objective 1. In particular, an outline of the architecture supporting the concept of Situational-Context (SC) has been presented. In [1], SC is defined as the composition of the virtual profiles of all entities involved in a given situation. The objective of a *situation* is to achieve the highest level of comfort possible for all participating entities. This is achieved by satisfying the greatest possible number of goals based on the available skills. Understanding by entities in a situation both: smart objects, and users represented through their smartphone.

According to [1], entities have two fundamental properties: *goals* and *skills*. Goals can be understood as an entity need that must be satisfied. For example, a user may need to turn on a lamp when a room is dark. Skills are those capabilities that can solve certain goals. For example, a lamp has the ability to increase the luminosity of a room. Thus, we work in a distributed environment, where the entities collaborate with each other to meet their goals.

Related to the above and to objective 3, the programming model aims to establish a new way of developing software for IoT systems. To do this, this model must be based on an architecture that allows defining the entities of a context. The architecture would be designed according to the Service-Oriented Computing paradigm. Figure 1 represents the components of the architecture:

- **Connectivity Manager:** Establishes the links between entities in a context. It sends and receives information from other entities by invoking services. This information is relative to goals, skills, history, etc. Communication between entities can be done using protocols like Bluetooth or WiFi. This is still undefined.
- **Entity Manager:** Entity's goals and skills will be defined, as well as its particular information, activity history or privilege level in the context. It will also have services that interpret data from the environment, through internal sensors, to be used in combination with the information provided by other entities when developing strategies.
- **History Service:** It stores the activity history. These activities are the result of interactions that have occurred in the past with other entities, why they interacted, when, where, what specific situation was being produced, etc. Thanks to this, further activities could be predicted.

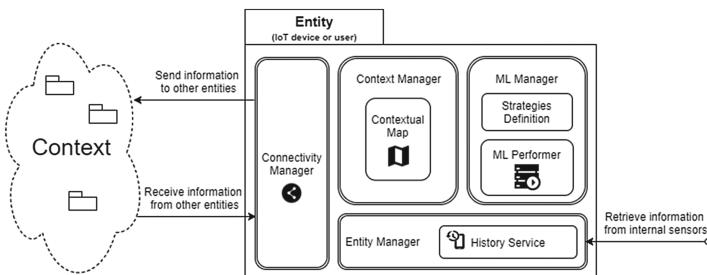


Fig. 1. Components of the proposed architecture

- **Context Manager:** It is responsible for creating/updating the contextual map. It will also be in charge of activating or deactivating strategies.
 - **Contextual Map:** Each entity will have its own contextual map based on the entities' goals and skills that surround it. So, each entity can know what goals can be solved with their skills. For instance, if a light has the ability to illuminate a room, it must be aware that it can do it.
- **ML Manager:** Covering the objective 2, it is responsible for the entity learning. Two types of learning can be differentiated. The first, individual, where the entity will learn from its own routines. The second, collective, in which the entity will learn from the entities that surround it within the context, with the aim of solving goals that their skills allow.
 - **Strategies Definition:** It analyzes the goals and skills to determine what goals can be addressed by the capabilities available in the contextual map. In addition, it must be considered that a goal could be solved using several different strategies.
 - **ML Performer:** It analyzes the activity history of the entity, allowing the system to learn and detect patterns of behavior, with the aim of automating tasks in the future.

From this architecture, software developers for IoT systems could work on new solutions. These solutions can use the goals and skills of different entities to provide new strategies to solve goals, or use the communication and learning mechanisms of the architecture, to add new skills to existing systems.

5 Evaluation Plan

Although this doctoral work is still in an initial state, we have developed some proofs of concept that validate SC in IoT environments. Besides, there are preliminary works to deal with the exchange of information at the network [3].

Our proposal may seem ambitious, however, as stated above, this work is still in an initial state. The full scope of the work will be discovered during its development. First, the architecture development. This task will focus on the development of services for the exchange of information between entities and strategies resolution for one and many users. We expect to achieve this in the second quarter of 2018. Second, a learning model, that will pursue executing ML algorithms in smartphones and to predict future actions based on the users' activities. It is planned that this task will provide valid results by the end of 2018. Next, the programming model. This will include a skills and abilities standardization through an ontology and using ML for strategies resolution. This task will be extended throughout 2019.

The case studies associated with these tasks will be developed under the 4IE project. A three years European project focused on gerontology [2].

In conclusion, the architecture and programming model that this thesis proposes will try to take full advantage of the features that IoT devices offer, while facilitating the task of developing software adaptable to them.

Acknowledgments. This work was supported by the Spanish Ministry of Science and Innovation (TIN2014-53986-REDT and TIN2015-69957-R), by the Department of Economy and Infrastructure of the Government of Extremadura (GR15098), and by the European Regional Development Fund and by 4IE project (0045-4IE-4-P) funded by the Interreg V-A España-Portugal (POCTEP) 2014–2020 program.

References

1. Berrocal, J., Garcia-Alonso, J., Canal, C., Murillo, J.M.: Situational-context: a unified view of everything involved at a particular situation. In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 476–483. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_34
2. Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: Rich contextual information for monitoring the elderly in an early stage of cognitive impairment. *Pervasive Mob. Comput.* **34**, 106–125 (2017)
3. Galán-Jiménez, J., Berrocal, J., Garcia-Alonso, J., Canal, C., Murillo, J.M.: Coordinating heterogeneous IoT devices by means of the centralized vision of the SDN controller (2017)
4. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. *IEEE Softw.* **31**(2), 48–53 (2014)
5. Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the internet of things to the web of things: resource-oriented architecture and best practices. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 97–129. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19157-2_5
6. Hirschfeld, R., Costanza, P., Nierstrasz, O.: Context-oriented programming. *J. Object Tech.* **7**(3), 125–151 (2008)
7. Keays, R., Rakotonirainy, A.: Context-oriented programming. In: Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 9–16. ACM (2003)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
9. Miranda, J., Mäkitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., Murillo, J.M.: From the internet of things to the internet of people. *IEEE Internet Comput.* **19**(2), 40–47 (2015)
10. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. *IEEE Commun. Surv. Tutor.* **16**(1), 414–454 (2014)
11. Roy, N., Misra, A., Cook, D.: Ambient and smartphone sensor assisted ADL recognition in multi-inhabitant smart environments. *J. Ambient Intell. Hum. Comput.* **7**(1), 1–19 (2016)
12. Schmidt, A.: Interactive context-aware systems interacting with ambient intelligence. *Ambient Intell.* **159** (2005)
13. Taivalsaari, A., Mikkonen, T.: A roadmap to the programmable world: software challenges in the IoT era. *IEEE Softw.* **34**(1), 72–80 (2017)



A Proposition for a Design Method of Service Systems

Blagovesta Kostova^(✉) 

École polytechnique fédérale de Lausanne, 1015 Lausanne, Switzerland
blagovesta.kostova@epfl.ch

Abstract. This research is in the domain of service science and proposes a method for designing service systems. We consider three levels of design – individual, organizational, and implementation. The proposed design method links these three levels by explicit input, output, and feedback between them. We use a design science research approach to design the method. This dissertation solves a practical question – how to design businesses that fit market needs, and has a theoretical contribution as it advances service science by combining theories from three domains: cognition (opportunity recognition), business modeling, and software engineering. We use systems thinking principles to federate the above theories. We gather data and validate our concepts in three contexts – a university course, startups, and established companies. We develop the necessary constructs and models, and conceptualize the final method in three iterations.

Keywords: Service design · Method · Opportunity recognition
Business model · Software engineering

1 State of Research

Service science studies complex socio-technical systems. To understand these systems, service science requires a multidisciplinary approach and a combination of methods and logic from various fields, such as computer science, psychology, design, and marketing [1]. We are interested in advancing service science by combining three research areas, namely: opportunity recognition, business modeling, and software engineering.

The desired outcome of this doctoral dissertation is a design method of service systems (Fig. 1) that links together three levels – an individual intuition of business, an organization of business, and an IT implementation. On the first design level, an individual, inspired by their environment, has business ideas that they believe to be valuable to a customer segment. In collaboration with other people, this individual designs the second (organizational) level to be a service system. The organizational level then feeds the third (IT implementation) level with fuzzy business needs. And the IT implementation phase yields a concrete artifact.

Under the supervision of professor Alain Wegmann.

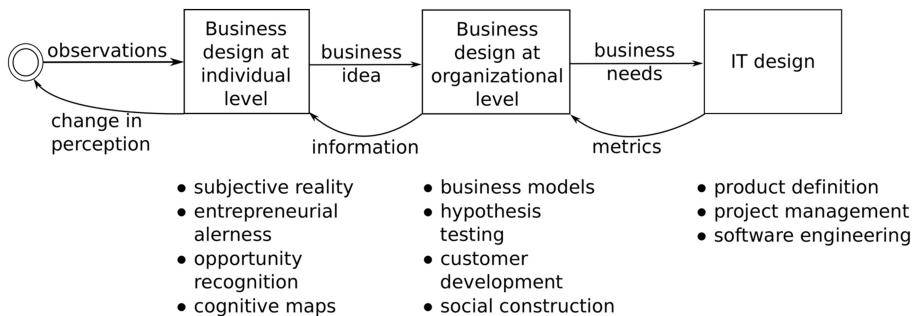


Fig. 1. Provisional service systems design method

On each level, there are the following flows:

- Input – information received from the previous step;
- Output – processed data provided to the next level;
- Feedback – meta information to guide the service designer in their work.

1.1 Individual Level

The literature describes opportunity recognition (identification) as a cognitive process that consists of active or passive search, alertness, and prior knowledge [2]. Tang et al. [3] argue that the most prominent traits for opportunity identification is entrepreneurial alertness. The pattern-recognition framework [2] describes the opportunity recognition phenomenon as entrepreneurs being able to draw parallels and find similarities (i.e., patterns) in various contexts, with an alertness that surpasses active search for information. Individuals use cognitive maps to internally represent the perceived information, and these maps link together seemingly unrelated notations into opportunities [4]. The last step is a reconfiguration of elements [5], which leads to an individual's proposal for a new reality to the society, hence, social construction.

Issues: The individual's intuition about a business opportunity does not translate flawlessly to the next level, where a group of people should achieve a shared understanding.

- Input: identification of observations, which leads to a business idea;
- Output: conceptualization of the observations of the individual, first level of structuring;
- Feedback: definition of the information the individual should seek to evaluate their perception regarding the business idea.

1.2 Organizational Level

This new reality goes beyond an individual's cognitive map and into an explicit shared understanding of what the envisioned reality would be. Entrepreneurs often use business models to communicate their business proposition with others. A business model captures the most important parts of a business – the way it creates and captures value

for a particular set of customers [6]. From a broad perspective, a business model is a story that explains how the enterprise works [7]. As a business model is an abstraction (it hides the complexity of implementation), the outcome of the implementation differs from this abstract description. We need feedback on the hypotheses in order to adjust the current situation and to be able to achieve the to-be state for the organization.

Issues: On the organization level, the transition between an individual's idea (an imagined service to deliver value) to a structured definition of the service system (conceptual shared service system design) is non-trivial.

- Input: an individual's cognitive maps;
- Output: a definition of a service system that considers individual cognitive maps.
- Feedback: a definition of heuristics and of metrics to be monitored from the implementation phase.

1.3 Implementation Level

The implementation level calls for an alignment between business and IT. Zachman [8] introduces an information-systems architecture that is foundational to the field of enterprise and service-oriented architecture. Weigand et al. [9] argue that to achieve alignment in enterprise architecture, we need to adopt a service perspective. IT-business alignment can be based on a transformation of values. For example, value-based software engineering recognizes the need for market justification information and communication technology (ICT) infrastructure [10]. This value-based view over software engineering serves as grounds for service-oriented modeling methods such as SEAM (Systemic Enterprise Architecture Methodology). SEAM is a family of methods for strategic thinking, business/IT alignment, and requirements engineering. SEAM is based on software engineering and on systems thinking philosophical principles [11].

Issues: On the implementation level, we need to define a desired input that corresponds to concrete business needs and is implementable (i.e., minimizing uncertainty). By tracing the business value, we need to be able to justify the software requirements. In addition, we need to be able to supply relevant metrics to the organization level in order to test business hypotheses.

- Input: mapping between the business needs and service requirements.
- Output: service that delivers a concrete IT artifact to service adopters.
- Feedback: data from the interaction to feed the feedback loops in the previous level.

1.4 Current Research

One existing research project in our area of research is on the alignment of human-centered service systems with corresponding business models [12]. This project focuses on designing principles in order to facilitate this alignment. It is a research project in its early stages; it has been presented as a research-in-progress. The goal of the project is to implement service innovations and their corresponding business models. The design principles guide service designers towards which actions to take. So far, two principles have been stated: define scenarios, and define scale and scope of the innovation.

The validation of the principles is ongoing. The project does not consider IT implementation. In addition, it features few details on the individual's cognition.

Hypothesis-testing entrepreneurship is based on the approach called Lean Startup [13]. It is a practice-oriented approach towards entrepreneurship. The presented work is a case that is summarized in [14]. This approach describes steps for achieving a product-market fit. Yet, the details on information and value exchange are minimal. The method is well-recognized in the industry but could be extended by data collection, analysis, and validation.

The current state of SEAM, the method for enterprise and service-oriented architecture, developed in our laboratory, includes models for analyzing and designing service-adopter motivation and the value-based alignment of hierarchical service systems. Our project uses SEAM to design service systems that connect the organizational and implementation levels, with a focus on value transformation between the levels. Potentially, we could extend SEAM to the individual designer's level and add explicit feedback mechanisms between levels.

2 State of Research Work Performed by Student

Our research involves three contexts: (1) on an individual level within a business design class for computer science students; (2) within the context of a collaborating startup, where we observe the transition from an individual level to an organizational level and to initial implementation; (3) within the context of an established company and implementation in a structured context:

- Teaching (individual): We describe our teaching approach that is based on experiential learning, systems thinking and service-dominant logic. Using repetition, we explain to our students how to recognize principles and patterns that exist in practice and how to apply the same principles and patterns in different contexts. During the 2017 semester, we collected data with two questionnaires. The project was presented as a research-in-progress at ISPIM 2017 [15]. We test our hypotheses about how individuals perceive business opportunities, how they structure their business ideas, and how they begin to model businesses.
- Startups (from individual to organization): We investigate how startups structures their business hypotheses, make decision-making process explicit, and evolve from an unstructured organization into a structured one. Our primary goal is to trace value exchange between elements in a business model, hence to design service systems based on value transformation. In this startup context, we collect data on how multiple individuals share their perceptions, form a shared view of the business idea, and shape the business needs into IT requirements.
- Industry (organization and implementation): We investigate the opportunity-recognition process within established companies. We collected data from a field project (four days of participation in requirements specifications workshops for a customer-relationship management system). We observe how a structured organization formalizes their business needs for new services.

3 Research Methodology: Design Science Research

The research methodology is based on design science research. “Design science [...] creates and evaluates IT artifacts intended to solve identified organizational problems” [16]. By designing an artifact, we solve a practical problem and contribute to the knowledge base. “Artifacts are defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)” [16].

In our case (Fig. 2), the practical problem is that startups struggle to design businesses and corresponding software solutions. Our artifact is a method for designing service systems. To build a method, first, we need to identify constructs and models, which are going to correspond to inputs, outputs, and feedbacks between the levels (individual, organization, implementation), and to evaluate their relevance. We perform a literature analysis to identify what already exists, and conduct field studies to understand what is in use. We evaluate them with expert interviews and case studies. Second, we design the artifact (a service design method) by using the previously identified set. The build phase is based on literature analysis and field studies. We evaluate to what extent the proposed method solves the practical problem with expert interviews, case studies, and formal verification of models. This concludes the application in the appropriate environment. Third, we contribute to the knowledge base. The evaluation criteria on the theoretical contribution come from the knowledge base methodologies.

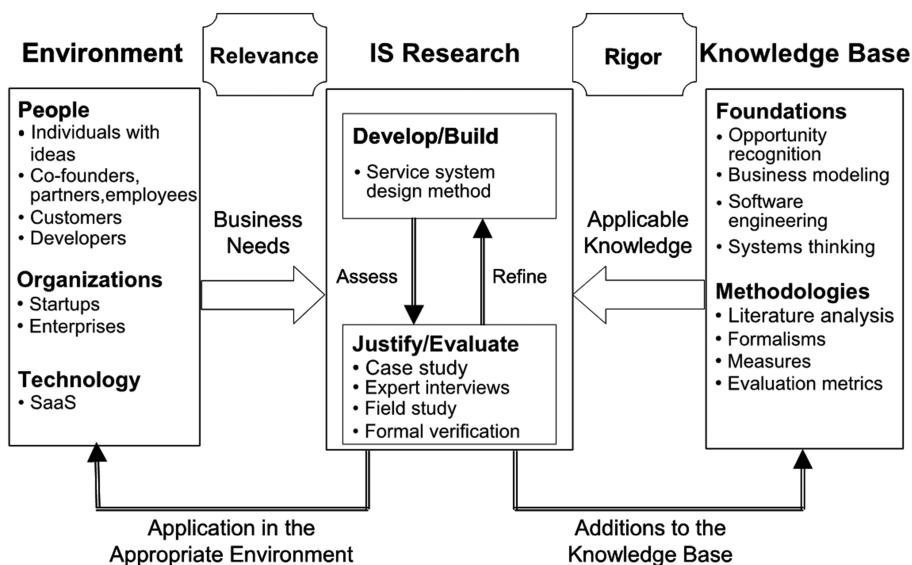


Fig. 2. Research design based on [16]

4 Doctoral Project Timeline

We follow an iterative approach. There will be three iterations, for each academic year, and a final thesis writing period after the last iteration (Table 1).

Table 1. Doctoral project timeline

Objective	Iteration 1	Iteration 2	Iteration 3
Constructs and models design	Sep–Nov 2017	Sep–Oct 2018	Sep 2019
Constructs and models evaluation	Dec 2017–Feb 2018	Nov 2018–Feb 2019	Oct–Dec 2019
Method design	Mar–May 2018	Mar–Apr 2019	Jan–Feb 2020
Method evaluation	Jun–Aug 2018	May–Aug 2019	Feb–Apr 2020

References

1. Maglio, P.P.: Editorial—Service Science 2.0. (2013)
2. Baron, R.A.: Opportunity recognition as pattern recognition: how entrepreneurs “connect the dots” to identify new business opportunities. *Acad. Manag. Perspect.* **20**, 104–119 (2006)
3. Tang, J., Kacmar, K.M.M., Busenitz, L.: Entrepreneurial alertness in the pursuit of new opportunities. *J. Bus. Ventur.* **27**, 77–94 (2012)
4. Weick, K.E.: Cartographic myths in organizations. *Mapp. Strateg. Thought* **1**, 1–10 (1990)
5. Gaglio, C.M., Katz, J.A.: The psychological basis of opportunity identification: entrepreneurial alertness. *Small Bus. Econ.* **16**, 95–111 (2001)
6. Chesbrough, H.: Business model innovation: opportunities and barriers. *Long Range Plann.* **43**, 354–363 (2010)
7. Magretta, J.: Why business models matter. *Harv. Bus. Rev.* **80**(5), 3–8 (2002)
8. Zachman, J.A.: A framework for information systems architecture. *IBM Syst. J.* **26**, 454–470 (1987)
9. Weigand, H., Johannesson, P., Andersson, B., Bergholtz, M.: Value-based service modeling and design: toward a unified view of services. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) *Advanced Information Systems Engineering, CAiSE 2009. Lecture Notes in Computer Science*, vol. 5565, pp. 410–424. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02144-2_33
10. Boehm, B.: Value-based software engineering. *ACM SIGSOFT Softw. Eng. Notes* **28**, 3 (2003)
11. Wegmann, A.: On the systemic enterprise architecture methodology (SEAM). In: *International Conference on Enterprise Information Systems* (2003)
12. Kleinschmidt, S., Burkhard, B., Hess, M., Peters, C., Leimeister, J.M.: Towards design principles for aligning human-centered service systems and corresponding business models. In: *Proceedings of the 37th International Conference on Information Systems*, pp. 1–11 (2016)
13. Eisenmann, T.R., Ries, E., Dillard, S.: *Hypothesis-Driven Entrepreneurship: The Lean Startup* (2012)
14. Reis, E.: *The Lean Startup*. Crown Business, New York (2011)

15. Kostova, B., Tapandjieva, G., Wegmann, A.: Teaching business design at an engineering school—principles/patterns/practice. In: ISPIM Innovation Symposium, p. 1. The International Society for Professional Innovation Management (ISPIM) (2017)
16. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**, 75–105 (2004)



A Model-Driven Approach to Continuous Delivery of Cloud Resources

Julio Sandobalin^{1,2(✉)}

¹ Escuela Politécnica Nacional, Ladrón de Guevara, E11-253,
P.O. Box 17-01-2759, Quito, Ecuador
julio.sandobalin@epn.edu.ec

² Universitat Politècnica de València, Camino de Vera, s/n,
46022 Valencia, Spain
jsandobalin@dsic.upv.es

Abstract. DevOps is a paradigm which brings practices and tools that optimize the software delivery time. Cloud-based DevOps processes facilitate the continuous delivery of infrastructure and software applications (i.e. cloud resources). In particular, Infrastructure as Code is the cornerstone of DevOps for automating the infrastructure provisioning based on practices from software development. There exist several Configuration Management Tools (CMTs) that use script languages to define the infrastructure provisioning to be deployed in a particular cloud provider. However, manual setting of the script languages to establish the infrastructure provisioning in a CMT for a particular cloud provider is a time-consuming and error-prone activity. For these reasons, the aim of my PhD research is proposing a model-driven approach to abstract and automate a continuous delivery process of cloud resources through model-driven techniques and DevOps. In addition, this approach seeks to cover the development process of cloud resources in development, testing and production environments.

Keywords: Cloud computing · DevOps · Continuous delivery
Infrastructure as code · Cloud resources · Model-Driven development

1 Introduction and Problem Statement

To succeed in a world where technologies, requirements, ideas, tools and timelines are constantly changing, information must be accurate, readily available, easily found and, ideally, constantly delivery in real-time to all members [1]. To automate the development software process practitioners are using a new paradigm called DevOps [2] (Development & Operations) which is promoting continuous collaboration between developers and operations staff through a set of principles, practices and tools that optimize the delivery software time. In particular, the cornerstone of DevOps is the Infrastructure as Code [3] which is an approach for automating the infrastructure provisioning based on practices from software development. There exist several cloud-based DevOps processes which leverage capacities offered by Cloud Computing and use the Infrastructure as Code for automating the infrastructure provisioning. Moreover, cloud-based DevOps processes facilitate the continuous delivery of

infrastructure and software applications (i.e. cloud resources). Configuration Management Tools (CMTs) such as Ansible¹, Puppet² or Chef³ have achieved automate the infrastructure provisioning in the Cloud. Each CMT has its script language to define the infrastructure provisioning to be deployed in a particular cloud provider. However, manual setting of the script languages to establish the infrastructure provisioning in a CMT for a particular cloud provider is a time-consuming and error-prone activity. Although CMTs have a high level of automation in the infrastructure provisioning, it remains a challenge to automate a model-based development process for continuous delivery of cloud resources.

This research is following the guidelines of the Design Science Methodology [4] (DSM) which is oriented to information system and software engineering. The goal of the DSM is obtaining an artifact in a problem context. Therefore, the artifact is:

A model-driven approach to continuous delivery of cloud resources.

The problem context is *cloud-based DevOps* and the stakeholders are *developers and operations staff*. To this end, I aim at answering the following research questions:

RQ1. Which DevOps-based approaches exist for continuous delivery of cloud resources?

RQ2. How can model-driven techniques support the automation of cloud infrastructure provisioning? **RQ2.1.** How to abstract the complexity to model the infrastructure of different cloud providers? **RQ2.2.** How to abstract the complexity to manage different script languages of the Configuration Management Tools?

RQ3. How to achieve a continuous delivery process of cloud resources based on models? **RQ3.1.** How to configure a DevOps toolchain for continuous delivery of cloud resources? **RQ3.2.** How to achieve a cloud resource development process based on models that cover the development, testing, and production environments?

2 Related Work

Currently, there is much interest in cloud-based DevOps research. Research efforts have focused on infrastructure provisioning and applications deployment in the Cloud. In this context, below are described the main research works that aim this approach.

TOSCA [5] is a standard for Topology and Orchestration Specification for Cloud Application which allows modeling nodes (virtual or physical machines) and orchestrates the deployment of Cloud applications. TOSCA uses DevOps provisioning tools such as Chef for infrastructure provisioning and Juju⁴ for deployment of cloud-based applications.

MORE [6] is a model-driven approach that focuses on automating of initial deployment and dynamic configuration of a system. MORE defines a topology of

¹ <https://www.ansible.com>.

² <https://puppet.com>.

³ <https://www.chef.io>.

⁴ <https://jujucharms.com>.

infrastructure to specify system structure and transforms this topology into executable code for Puppet tool to get virtual machines, physical machines and containers.

MODAClouds [7] is a European project undertaken to simplify the cloud services usage process. One of its goals is to support system developers in building and deploying applications and related data to multi-clouds spanning across the full cloud stack. MODAClouds includes automated infrastructure provisioning platform using Puppet's modules.

On the other hand, research works that provide guidelines for continuous delivery have focused their efforts on code-based software delivery process. The main approaches in this context are following.

Soni [8] present a research work that focuses on the necessities of the insurance industry. This approach proposes a proof of concept for designing an effective framework for continuous integration, continuous testing, and continuous delivery to automate the source code compilation, code analysis, test execution, packaging, infrastructure provisioning, deployment and notifications using build pipeline concept.

Rathod and Surve [9] propose a framework for automated testing and deployment to help automated code analysis, test selection, test scheduling, environment provisioning, test execution, results from analysis and deployment pipeline.

3 Proposed Solution

Answering the research question ***RQ2***, to support the automation of cloud infrastructure provisioning through model-driven techniques I have developed ARGON [10] (*An infRastructure modelinG tool for clOud provisioNing*) tool. ARGON has two main components: (a) Domain Specific Language (DSL) to model the infrastructure in the Cloud (***RQ2.1***), and (b) Transformation engine which creates scripts to manage different Configuration Management Tools (***RQ2.1***).

ARGON has an abstract syntax which is defined through an *Infrastructure Metamodel* [10] (see Fig. 1a). The metamodel abstract the capacities of the cloud computing instead of focus on infrastructure provisioning tools. Thus, I can distinguish four groups according to the cloud capacities: (1) ***Computing*** capacity allows the creation of *Virtual Machines* with one or more *Security Groups* that perform as a firewall. Each *Security Group* enables a *Virtual Machine* access through ports as *Inbound* rules and *Outbound* rules. *Load Balancer* allows distributing incoming application traffic between multiple *Virtual Machines* and with an input rule or *Listener* checks the connection requests. In addition, I can assign a *Static IP* address to a *Virtual Machine*. (2) ***Storage*** capacity allows the creation of *Databases* and *File* servers. (3) ***Elasticity*** capacity allows the creation of templates or *Launch Configuration* where features of a *Virtual Machine* are specified. Templates are used to configure the creation of groups of *Virtual Machines* by means of *Auto Scaling Group*. Creation or elimination of *Virtual Machines* is done based on *Scaling Policy* which is executed by an *Alarm* that monitor a metric in a period of time. (4) ***Networking*** capacity is represented by associations among metaclasses.

The metamodel only defines the abstract syntax, but not a concrete notation of the graphical language in ARGON. In order to use graphical notation to render the model

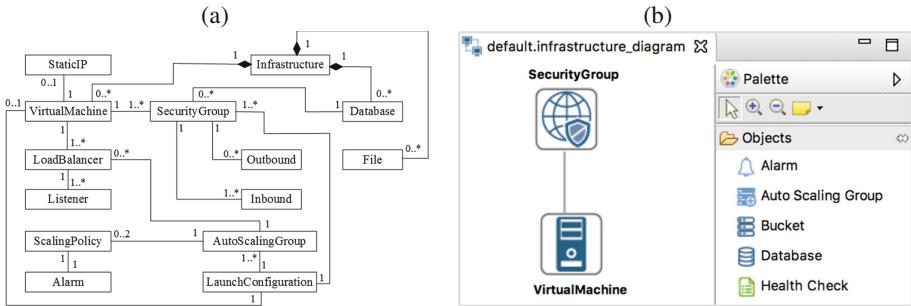


Fig. 1. (a) Infrastructure Metamodel. (b) Infrastructure Model.

elements in the modelling editors, I use a concrete syntax developed by using EuGENia [11]. EuGENia facilitates to generate the models needed to implement a GMF editor in Eclipse Modeling Framework [12]. ARGON uses this DSL to create an *Infrastructure Model* [10] (see Fig. 1b) representing the infrastructure with its provisioning requirements of hardware and software.

The transformation engine creates configurations files or scripts that have the full instructions to create hardware and its settings and install the underlying software. The transformation engine abstracts the features of scripting languages of Configuration Management Tools (CMTs) to create transformation rules which represent modules to build cloud elements. The transformation engine uses an infrastructure model to apply on it the model-to-text transformations and generates scripts for CMTs. It is worth mentioning that both DSL and transformation engine are JARs (Java ARchives) which can be used with the Eclipse packages.

On the other hand, as a first approach to answering the research question *RQ3*, I have configured an end-to-end automated toolchain for infrastructure provisioning in the Cloud based on DevOps community tools [13] (*RQ3.1*) and ARGON [10]. This approach takes advantage of Infrastructure as Code concept to apply DevOps practices by supporting to a systematized and automated *infrastructure provisioning pipeline* [13] (see Fig. 2) (*RQ3.2*). I use ARGON tool to model an infrastructure model with its provisioning requirements of hardware and software. Subsequently, I will take this infrastructure model and push it toward a *Version Control* system in order to retain and provide access to every version of every infrastructure model that has ever been stored on it. Moreover, this approach allows teams with infrastructure models across different places to work collaboratively. An *Artifact Repository* is used to provide software libraries, such as the model-to-text transformation engine and ARGON's Domain Specific Language. I use an artefact repository in order to provide an only one provider of libraries or software artefact in phases of development, build, and testing.

Every infrastructure model must be checked into a single version control repository to begin the *Continuous Integration* stage. Continuous integration requires that every time developers or operations staff commits a change, the entire application is built and a compressive set of automated tests run against it [2]. The transformation engine is used as a plugin in the continuous integration server to create scripts for provisioning tools. After, a set of the automated test proposed by Morris [3] is run against the scripts.

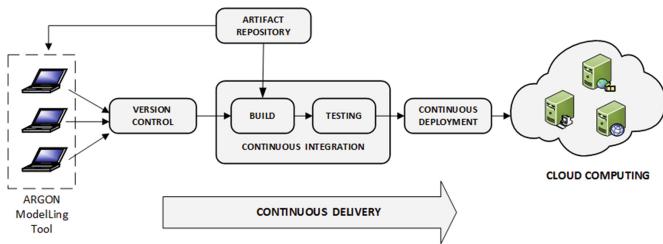


Fig. 2. Overview of the infrastructure provisioning pipeline.

First, *Syntax Check Tests* are executed for the verification of the structure of the scripts. Second, *Static Code Tests* are performed by parsing code or definition files without executing them in the Cloud.

Scripts that have been built and have overcome a set of automated tests are ready to be used in CMTs. *Continuous Deployment* stage takes these scripts built in the previous stage and automatically use them in CMTs to orchestrate the infrastructure provisioning and software deployment in the Cloud. Finally, once the infrastructure provisioning is finished, the *Infrastructure Tests* are executed towards ensuring the correct functioning of the infrastructure and the software deployed in the Cloud.

4 Conclusions and Future Directions

At the end of the second year of my PhD research⁵, my work has focused on demonstrating the feasibility of how to support the continuous delivery of cloud resources through model-driven techniques and DevOps. First, ARGON tool provides support to model the cloud infrastructure elements and then generate scripts to manage Configuration Management Tools. Second, an end-to-end automated DevOps toolchain brings support for continuous delivery of infrastructure in the Cloud based on infrastructure models developed by ARGON tool.

The next step of my PhD research⁶ is to design a first experiment to validate the ARGON tool. The propose of the experiment is to obtain the effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use the ARGON tool from the point of view of software engineering, in the context of undergraduate and post-graduate students in Computer Science. On the other hand, ARGON tool provides support for Amazon Web Services. Thus, I am going to extend the ARGON tool to support infrastructure modeling for different cloud providers such as Microsoft Azure and Google Computing Engine. Additionally, I am going to extend the DevOps toolchain to support the continuous delivery process for different cloud providers. Finally, I am going to generalize the findings for proposing a model-driven method for continuous delivery of cloud resources.

⁵ I started my PhD research in September 2015.

⁶ The next academic year 2017/2018 is the third year of my PhD in which I have to finish my research.

Acknowledgments. This research is supported by the Value@Cloud project (TIN2013-46300-R).

References

1. Cois, C.A., Yankel, J., Connell, A.: Modern DevOps: optimizing software development through effective system interactions. In: IEEE International Professional Communication Conference (IPCC) (2015)
2. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 1st edn. Addison-Wesley Professional, Upper Saddle River (2010)
3. Morris, K.: Infrastructure As Code: Managing Servers in the Cloud, 1st edn. O'Reilly Media Inc, Sebastopol (2016)
4. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Springer, Heidelberg (2014). <https://doi.org/10.1007/978-3-662-43839-8>
5. Wettinger, J., Breitenbürger, U., Kopp, O., Leymann, F.: Streamlining DevOps automation for Cloud applications using TOSCA as standardized metamodel. Future Gener. Comput. Syst. **56**, 317–332 (2015)
6. Chen, W., et al.: MORE: a model-driven operation service for cloud-based IT systems. In: Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016, pp. 633–640 (2016)
7. Di Nitto, E., Matthews, P., Petcu, D., Solberg, A.: Model-Driven Development and Operation of Multi-Cloud Applications. Springer, Cham (2017). <https://doi.org/10.1007/978-3-319-46031-4>
8. Soni, M.: End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery. In: Proceedings - 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2015, pp. 85–89 (2016)
9. Rathod, N., Surve, A.: Test orchestration a framework for Continuous Integration and Continuous deployment. In: 2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015 (2015)
10. Sandobalin, J., Insfran, E., Abrahao, S.: An infrastructure modelling tool for cloud provisioning. In: Proceedings - 14th IEEE International Conference on Services Computing, SCC (2017)
11. Kolovos, D.S., García-Domínguez, A., Rose, L.M., Paige, R.F.: Eugenia: towards disciplined and automated development of GMF-based graphical model editors. Softw. Syst. Model. **16**(1), 229–255 (2015)
12. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional, Lebanon (2008)
13. Sandobalin, J., Insfran, E., Abrahao, S.: End-to-End automation in cloud infrastructure provisioning. In: Proceedings - 26th International Conference on Information Systems Development, ISD (2017)



SLA-Driven Governance for RESTful Systems

Antonio Gamez-Diaz^(✉), Pablo Fernandez, and Antonio Ruiz-Cortes

Universidad de Sevilla, Seville, Spain
agamez2@us.es

Abstract. Software distribution models are moving to SaaS paradigms where customers no longer need to buy a perpetual license. In this context, SaaS providers leverage the Service Level Agreement (SLA) concept to delimit the functionality and guarantees to which they commit to their customers. However, although formal specifications for the definition of SLAs have been proposed, providers usually have an ad-hoc approach with a low degree of automation. This approach confirms the fact that the SaaS industry has not incorporated the idea of an SLA model that can be implemented within the infrastructure as a decision mechanism. This instrumentation would be of special interest in RESTful microservice architectures in providing an automated governance framework for the service catalog and regulating the behavior of each component in the context of the agreements reached with each client.

This thesis project is divided in four stages: (i) Establishing a sufficiently expressive specification for the description of RESTful microservices regulated by advanced SLAs; (ii) Develop a catalog of SLA analysis and management operations to support the governance of micro-service architectures; (iii) Implement a SLAs management ecosystem to support the government of RESTful microservices; (iv) Consolidation of the Governify platform to validate the proposal in industrial environments.

1 State of the Art

The *Software as a Service* (SaaS) paradigm has become entrenched in the industry as a deployment model, bringing flexibility to the customer in pay-for-use models and making unnecessary to maintain its own infrastructure. In particular, these types of systems present a business model that depends on the expected benefits for certain infrastructure costs.

The main architectural paradigm of SaaS systems are the service-oriented architectures (SOA) [5, 10]. They have a number of characteristics such as low coupling, abstraction, reusability, autonomy, and interoperability. These architectures have traditionally relied on a set of specifications (SOAP, WSDL) that

This thesis project is being partially supported by the FPU scholarship program, granted by the Spanish Ministry of Education, Culture and Sports (FPU15/02980) and the European Commission (FEDER), the Spanish and the Andalusian R&D&I programs (grants TIN2015-70560-R (BELI) and P12TIC-1867 (COPAS)).

allow varying degrees of granularity and provide complex interfaces that often require heavy deployment infrastructures.

In recent years, there has been a trend towards a new architectural style that has been called microservice. This style requires that each component (a microservice) can evolve, scale and deploy independently to the rest, increasing the flexibility of the system as a whole. This architectural style is employed by high-performance commercial systems architecture such as eBay, Amazon and Netflix [8].

A common element of these architectures in particular, and, in general, when defining and implementing microservices is that they follow the RESTful paradigm. This paradigm provides a unified approach to identify the granularity and operational interface of microservices that has a high degree of extensibility. In particular, RESTful provides a lighter approach for building, deploying and cloning microservices more effectively.

RESTful systems provide numerous advantages in terms of elasticity, fault tolerance and flexible architectures design [3]. Both industry [7] and academia [1] agree to identify the management and evolution of services as a key element to achieve a more agile integration of the systems and to have a technological infrastructure that responds quickly to the business needs. These challenges are magnified in the context of microservice architectures since the independent life cycle of each microservice must be coordinated as part of a service catalog that has a higher growth rate than that of classical architectures [9].

As a first step in solving this problem, with a similar approach that in other industries, SaaS providers use the Service Level Agreement (SLA) concept to delimit the functionality and guarantees to which they commit to their customers. However, although formal specifications have been proposed for the definition of SLAs that arise in the context of the classic SOAs (WS-Agreement [2]), providers usually have an ad-hoc approach with a low degree of automation. This approach confirms the fact that the SaaS industry has not incorporated the idea of an SLA model that can be implemented within the infrastructure as a decision mechanism when it comes to supporting the SaaS business model. This instrumentation would be of special interest in RESTful microservice architectures in providing an automated governance framework for the service catalog and regulating the behavior of each component in the context of the agreements reached with each client.

2 Research Challenges

The thesis project focuses on the study of the ways to improve and automate the management of microservice architectures based on RESTful systems regulated by SLAs through the definition of methodologies, techniques and tools. Specifically, we have identified four main challenges:

- D1: Establishing a sufficiently expressive specification for the description of RESTful microservices regulated by advanced SLAs

This challenge would be focused on defining a specification that has the necessary elements to support two main requirements: (i) Establishing the service properties necessary to regulate government infrastructures that allow aligning the behavior of the system with the business model; in this way, these properties would guide the automated decision making in the different components of the architecture (e.g. their scalability strategies); (ii) Defining a price model that has sufficient degree of expressiveness to contemplate the actual variability of the purchasing processes taking into account aspects such as discounts or compensation [6].

From the research context point of view, this challenge would represent the extension of the industrial specification for RESTful systems most used (OAI) to meet the requirements presented.

- D2: Develop a catalog of SLA analysis and management operations to support the governance of micro-service architectures

This challenge would address the characterization of the requirements to support the government of real SaaS scenarios regulated by SLAs with micro-service architectures. In particular, these scenarios can be classified into two types: (i) End-user information systems containing a backend-frontend (applications) schema; (ii) Integrated information systems that have only one backend component (APIs).

These two typologies contain specific requirements to be characterized as part of the challenge in order to define a complete set of SLA analysis and management operations that allow the extraction of adequate information to support automated decision making in the provision of services and definition of business strategies.

- D3: Implement a SLAs management ecosystem to support the government of RESTful microservices

Based on the catalog identified in D2, this challenge aims to address the evolution of current SLAs management libraries to an ecosystem scheme articulated in microservices as new components within the Governify platform. This ecosystem would allow integrating SLAs defined based on the specification proposed in D1 to achieve an automated SaaS instrumentation with independent domain tools.

Another interesting challenge to face is about how to automatically derive SLA profiles of services by observing past performance and learning their behaviors. With this approach, companies can leverage from SLA-governance solutions without the associated costs of specifying every metric threshold.

- D4: Consolidation of the Governify platform to validate the proposal in industrial environments

Since Governify has been designed to avoid ad-hoc approaches in SaaS governance, it has been designed as an open set of components that define a unified and homogeneous interface to foster the development of new research and industrial ICT projects.

3 Preliminary Results

In particular, during the first thesis year we have addressed the contribution C1, giving as a result the following activities:

- **C1.1: Technical workshops** dealing with the technological ecosystem that is closely related to the thesis topic: (i) *REST APIs Security* invited talk for master students; (ii) *Docker* and *Kubernetes* workshop for master students. (iii) Technical meetings of *servitization* given to nodes of the RCIS network in order to share the research results with other nodes.
- **C1.2: Conferences**, both national and international: **C1.2.1 Towards SLA modeling in RESTful APIs**, presented at the JCIS national conference; **C1.2.1 An analysis of RESTful APIs offerings in the industry**, accepted at 15th ICSOC edition.
- **C1.3: Journals:** we are preparing a journal extension for the paper accepted on ICSOC as well as we are incepting another one on the specification of SLAs for RESTful APIs.

4 Relevance

This thesis project extends the scope of *BELI*, a Spanish national government project, whose objectives include the development of a governance system for hybrid services systems that integrate information systems (computer services) and teams of people. In this context, the initial stages have been covered with the Governify¹ ecosystem, which provides a set of microservices that support the design, monitoring and implementation of SLAs.

The objectives of the project are closely aligned with the IMPACT (H2020) European project, which aims to develop a set of techniques and tools to support the evolution of information systems to cloud platforms in the context of public administrations. In this context, the present thesis project will try to provide the instrumentation mechanisms that can be incorporated in the integration of public cloud APIs to verify the fulfillment of the providers guarantees and the regulations required by public administrations.

The relevance in the industry of the thesis topic is evidenced in the numerous ICT projects. Specifically, we highlight *SLA@OAI* project, developed in conjunction with *Icinetic*, which aims to develop an operational extension to define SLAs within the framework of the Open API Initiative (OAI) specification², which currently represents the reference standard, widely used in the industry, to define functional interfaces of RESTful services.

Furthermore, several European projects have been framed by this idea of service governance (SLALOM, SLA-Ready, CELAR, S-CUBE, RESERVOIR and SLA@SOI). These projects approach the topic of the Service-Oriented Applications (SOA) governance from different perspectives, but all of them place SLAs as a key element for the advanced management of infrastructure [4].

¹ <http://www.governify.io/>.

² <https://www.openapis.org/>.

References

1. Aier, S., Offermann, P., Schönherr, M., Schröpfer, C.: Implementing non-functional service descriptions in SOAs. In: Draheim, D., Weber, G. (eds.) TEAA 2006. LNCS, vol. 4473, pp. 40–53. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75912-6_4
2. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical report, Open Grid Forum (2004)
3. Costa, B., Pires, P.F., Delicato, F.C., Merson, P.: Evaluating REST architectures-approach, tooling and guidelines. *J. Syst. Softw.* **112**, 156–180 (2014)
4. Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., Pohl, K.: A journey to highly dynamic, self-adaptive service-based applications. *Autom. Softw. Eng.* **15**(3–4), 313–341 (2008). Special issue
5. Georgakopoulos, D., Papazoglou, M.P.: Service-Oriented Computing. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-10383-4>
6. Kansal, S., Singh, G., Kumar, H., Kaushal, S.: Pricing models in cloud computing. In: Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies, ICTCS 2014, pp. 1–5. ACM Press, New York (2014)
7. Kavianpour, M.: SOA and large scale and complex enterprise transformation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 530–545. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74974-5_50
8. Mauro, T.: Microservices at Netflix: Lessons for Architectural Design
9. Newman, S.: Building Microservices. O'Reilly Media, Sebastopol (2015)
10. Papazoglou, M.P., van den Heuvel, W.-J.: Service oriented architectures: approaches, technologies and research issues. *VLDB J.* **16**(3), 389–415 (2007)



Towards Adaptive Monitoring Services for Self-Adaptive Software Systems

Edith Zavala^(✉)

Universitat Politècnica de Catalunya (UPC), Jordi Girona, 1-3, 08034 Barcelona,
Catalunya, Spain
zavala@essi.upc.edu

Abstract. In order to deal with the great diversity of execution contexts, modern software systems rely on feedback control loops and external monitoring services for observing them and their environment, and respond to context changes through adaptation. In this process, the monitoring services play a crucial role since the quality of the monitoring data (e.g., timeliness, freshness, accuracy, availability, etc.) affects directly the self-adaptation decisions. Most of the current approaches supporting monitoring for self-adaptive systems (SAs) assume that the monitors are static components and they do not change at runtime. Due to the dynamism of execution contexts mentioned before, this vision is not valid anymore. Nowadays, monitoring services need to be adaptive as well, in order to respond to context changes, e.g., new measures to collect are required or a monitor service failure occurs at runtime. The complexity of handling monitoring services adaptation in coordination with SAs operation challenges and offers new opportunities to software engineers. In order to address this challenge, this research proposes to extend the adaptation logic of modern SAs with an external MAPE-K loop for managing the adaptation process of the monitoring services participating in the SAs adaptation logic. Different algorithmic, statistical, modeling and stochastic analysis and decision-making techniques are being explored for implementing this loop. Moreover, a reusable architecture for enacting the adaptation decisions in the monitoring services is being developed. The approach will be evaluated in two systems: a self-adaptive smart vehicle and a real self-adaptive video streaming service.

Keywords: Adaptive monitoring services · Self-adaptive systems
Runtime adaptation

1 Introduction

The complexity of modern software systems has increased dramatically over the years and is continuing to do so [1]. Users can access software applications using a variety of devices and since mobile technologies are on the rise, applications are becoming

Supervised by Prof. Xavier Franch and Dr. Jordi Marco.

ubiquitous in our society. In order to deal with the great diversity of execution contexts, different user profiles, system faults, changing environmental conditions and user needs, etc., modern software systems provide self-adaptation capabilities at runtime.

In practice, these capabilities are usually realized by implementing the prominent MAPE-K feedback control loop [2, 3]. In MAPE-K, self-adaptation is achieved applying four steps: Monitor, Analyze, Plan, and Execute. In the Monitor step, runtime data from the targeted adaptive system and its environment is collected. The collected data is then Analyzed and, if needed (e.g. analysis results show a violation of the requirements), a system adaptation is Planned. Finally, the adaptation is enacted in the Execute step. In order to function properly, these four steps share a Knowledge base containing data such as measurements, logs, adaptation policies, etc.

Modern SASs rely on external monitoring services for observing them and their environment. In the MAPE-K loop, the monitoring services play a crucial role since the quality of the monitoring data (e.g., correctness, timeliness, freshness, accuracy, availability, etc.) affects directly the performance of the rest of the elements of the loop and in consequence the self-adaptation decisions made [2].

In order to guarantee the quality of the monitoring data at runtime, monitoring services supporting SASs have to be adaptive as well. The complexity of handling the monitoring services adaptation in coordination with SASs operation challenges and offers new opportunities to software engineers. In this research, alternatives for addressing this challenge are studied. The rest of the paper is organized as follows. Section 2 states the problem this research intends to solve. Section 3 details the proposed solution. Section 4 presents related work. And finally, Sect. 5 presents conclusions and future work.

2 Problem Statement

Several approaches have been proposed to support monitoring of SASs and their environment. However, most of them assume that the Monitor element (as well as the rest of elements) of the loop is a static component and it does not change at runtime. This implies that the systems' owners should know everything to be monitored at design time and that the monitoring services that compose this MAPE-K element always behave the same. Due to the systems' changing environment, monitoring services dynamism and evolving user and systems' owners' requirements, this vision is not valid anymore.

As mentioned before in Sect. 1, monitoring services supporting modern SASs need to be adaptive as well in order to guarantee monitoring data quality (e.g., correctness, timeliness, freshness, accuracy, availability, etc.) and the requirements' satisfaction over time. The challenge of handling the monitoring services adaptation process in coordination with SASs operation motivates the study, specification, design and development of new and innovative methods and techniques for effectively, efficiently and accurately planning and applying adaptation decisions on monitoring services supporting SASs at runtime.

Motivated by this challenge, this research explores and experiments with new and existing methods and techniques in order to provide a solution to deal with the

automatic adaptation of the monitoring services supporting MAPE-K-based SASs. In summary, the research addresses two main objectives:

- Enable the automatic adaptation of monitoring services supporting MAPE-K-based SASs at runtime in order to respond to changes in the environment and the monitoring services themselves.
- Compare the results of using adaptive monitoring services in MAPE-K-based SASs, against using non-adaptive (static) monitoring services.

In order to meet these research objectives we have to address the following research questions:

- **RQ1.** How is monitoring adaptation supported by existing approaches?
- **RQ2.** How is the monitoring services adaptation at runtime supported in current MAPE-K-based SASs by existing approaches?
- **RQ3.** What can be improved or done for better supporting the adaptation of the monitoring services at runtime in MAPE-K-based SASs in the existing approaches?

3 Proposed Solution

In order to coordinate the adaptation of the monitoring services conforming the Monitor element of MAPE-K-based SASs, this research proposes: first, extend the IBM MAPE-K loop architecture [3] utilized by most of the modern SASs for conducting their adaptation process, with a second MAPE-K loop on top of the Monitor element in charge of managing the adaptation process of the monitoring services (see Fig. 1). This design decouples the monitoring services adaptation logic from the SASs adaptation logic, allowing the independent development of the modules and the adoption of the approach by existing SASs.

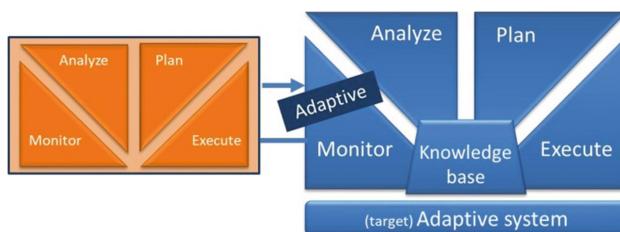


Fig. 1. Adaptive monitoring services for SASs (Color figure online)

Second, we propose to explore different methods and techniques identified in **RQ1** and **RQ2** for implementing the MAPE-K loop in charge of the monitoring services adaptation. Then, determine which of them perform better for monitoring services supporting SASs. A flexible and generic design of the external MAPE-K loop elements (orange elements in Fig. 1) will be required for experimenting with the different techniques. At this point, we are exploring different algorithmic, statistical, modeling

and stochastic analysis and decision-making techniques, through a systematic mapping in the domain of adaptive monitoring. Finally, we are developing a reusable high-level architecture for adaptive monitoring services participating in MAPE-K loops. The architecture is intended to serve as a reference for correctly managing the enactment process of monitoring adaptation decisions at runtime.

We plan to evaluate our solution in two systems: a self-adaptive smart vehicle and a real self-adaptive video streaming service, this second as part of the SUPERSEDE (SUpposing evolution and we adaptation of PERsonalized Software by Exploiting contextual Data and End-user feedback) H2020 research and innovation project (<https://www.supersede.eu>). In the first scenario, the monitoring services adaptation aims at providing energy-efficient monitoring for self-adaptive smart vehicles. In the second scenario, adaptation is performed in order to guarantee monitoring services resilience for the self-adaptive video streaming service. Techniques for implementing the external loops of both systems (orange elements in Fig. 2) are being studied.

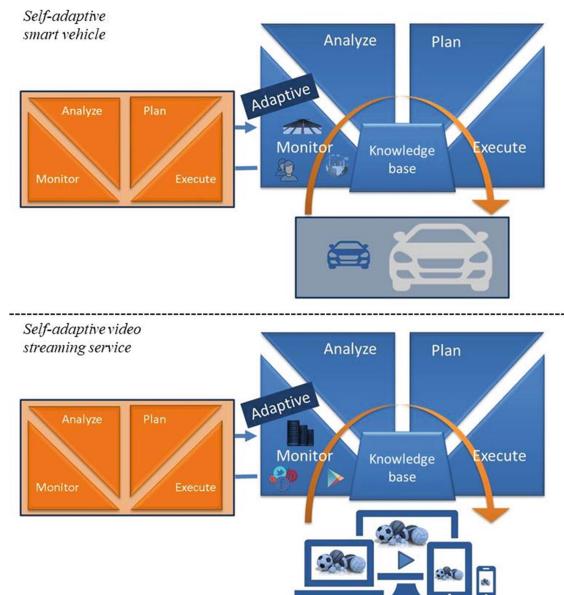


Fig. 2. Adaptive monitoring services for SASs evaluation (Color figure online)

Regarding the evaluation, we have already results about the adaptation of the target systems with static monitoring. For the smart vehicle a preliminary version of the implementation has been presented as a demo tool in the IEEE RE'15 conference [4]. On the other hand, the advances on the video streaming service are available in the public deliverable D4.5 of the SUPERSEDE project [5].

4 Related Work

So far, a lot of research has been performed on adapting monitoring systems in a diversity of domains, for instance, e-health monitoring services [6, 7], structural and environmental monitoring hardware and software systems [8, 9], network probing [10, 11] and software systems monitoring applications [12, 13]. In this work, due to space limitations, we reference only some of the studies dealing with the adaptation of monitoring systems; however, we know that there are several more since we are conducting a systematic mapping in which we are analyzing a final set of 115 papers related to this topic.

Although lot of effort has been done, only very few of existing works consider the adaptation of monitoring systems that support SASs. And in consequence the challenge of handling the adaptation process of the monitors in coordination with the normal operation of SASs is still open. On the other hand, the importance of making the MAPE-K loop elements adaptive has been remarked in the last years by other works [2, 14, 15]. However, the proposals are very superficial or designed for solving a particular problem. Unlike existing work, we propose to develop a reusable solution to place on top of any type of monitoring service in the adaptation logic of a SAS.

5 Conclusions and Future Work

In this paper details about our ongoing research on the adaptive monitoring service for MAPE-K-based SASs topic have been provided. We stated the motivation of the research and the research problem and challenge it tries to solve. Moreover, we described our initial proposal for addressing the research challenge as well as the systems in which we plan to evaluate this proposal. We have also presented the related work and pointed out the novelty of our research regarding existing proposals. For the advances on the research results, we have initial implementations of the self-adaptive smart vehicle and video streaming service with static monitoring services. The next step is to provide adaptation capabilities to these monitoring services applying our proposed solution and compare evaluation results using the static and the adaptive monitoring services in the same systems (i.e., address **RQ3**).

Acknowledgments. Thanks to CONACYT and I2T2, for the PhD scholarship granted to Edith Zavala. This work is partially supported by the SUPERSEDE project, funded by the European Union's Information and Communication Technologies Programme (H2020) under grant agreement no. 644018 and partially funded by the Spanish project GENESIS (TIN2016-79269-R).

References

1. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A survey on engineering approaches for self-adaptive systems. *Pervasive Mob. Comput.* **17**, 184–206 (2015). <https://doi.org/10.1016/j.pmcj.2014.09.009>

2. Toueir, A., Broisin, J., Sibilla, M.: A goal-oriented approach for adaptive SLA monitoring: a cloud provider case study. In: 2nd IEEE Latin American Conference on Cloud Computing and Communications (LatinCloud), pp. 53–58. IEEE (2013)
3. IBM-Corporation: An architectural blueprint for autonomic computing. IBM White Paper 36, 34 (2006). <https://doi.org/10.1021/am900608j>
4. Zavala, E., Franch, X., Marco, J., Knauss, A., Damian, D.: SACRE: a tool for dealing with uncertainty in contextual requirements at runtime. In: 23rd IEEE International Requirements Engineering Conference (RE), pp. 278–279. IEEE (2015)
5. Gorroñogoitia, J., Zavala, E., Oriol, M., Motger, Q., Stevanetic, S.: D4.5: methods and tools to enact software adaptation and personalization, v2, Deliverable D4.5 of SUPERSEDE H2020 project (2016)
6. Mshali, H.H., Lemlouma, T., Magoni, D.: Context-aware adaptive framework for e-health monitoring. In: IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), pp. 276–283. IEEE, USA (2016)
7. Fan, L., Xiong, L.: Real-time aggregate monitoring with differential privacy. In: 21st ACM International Conference on Information and Knowledge Management (CIKM), p. 2169. ACM Press, New York (2012)
8. Kim, S., Pakzad, S.: Wireless sensor networks for structural health monitoring: a multi-scale approach. In: 17th Analysis and Computation Specialty Conference at Structures Congress (ASCE) (2006)
9. Iacono, M., Romano, E., Marrone, S.: Adaptive monitoring of marine disasters with intelligent mobile sensor networks. In: IEEE Workshop on Environmental Energy and Structural Monitoring Systems (EESMS), pp. 38–45. IEEE (2010)
10. Shen, D., Tse, K.H., Chan, C.K.: Adaptive fault monitoring in all-optical networks utilizing real-time data traffic. *J. Netw. Syst. Manag.* **20**, 76–96 (2012)
11. Shamsi, J., Brockmeyer, M.: Predictable service overlay networks: predictability through adaptive monitoring and efficient overlay construction and management. *J. Parallel Distribut. Comput.* **72**, 70–82 (2012). <https://doi.org/10.1016/j.jpdc.2011.09.005>
12. Psiuk, M., Zielinski, K.: Goal-driven adaptive monitoring of SOA systems. *J. Syst. Softw.* **110**, 101–121 (2015). <https://doi.org/10.1016/j.jss.2015.08.015>
13. Gonzalez-Herrera, I., Bourcier, J., Daubert, E., Rudametkin, W., Barais, O., Fouquet, F., Jézéquel, J.M., Baudry, B.: ScapeGoat: spotting abnormal resource usage in component-based reconfigurable software systems. *J. Syst. Softw.* **122**, 398–415 (2016). <https://doi.org/10.1016/j.jss.2016.02.027>
14. Roth, F.M., Krupitzer, C., Becker, C.: Runtime evolution of the adaptation logic in self-adaptive systems. In: 12th IEEE International Conference on Autonomic Computing Runtime, pp. 141–142 (2015)
15. Klos, V., Gothel, T., Glesner, S.: Adaptive knowledge bases in self-adaptive system design. In: Proceedings of 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015, pp. 472–478 (2015). <https://doi.org/10.1109/seaa.2015.48>



An Approach to Predictive Analysis of Self-Adaptive Systems in Design Time

Patrícia Araújo de Oliveira^(✉), Francisco Durán, and Ernesto Pimentel

University of Málaga, Málaga, Spain
{patricia,duran,ernesto}@lcc.uma.es

Abstract. Predictive analysis methods offer the possibility of estimating the impact of design decisions, which may help in the accomplishment of operational optimal results, before the deployment of the system is made, and therefore minimizing the required maintenance effort and cost. However, current predictive methods are not effective when used on self-adaptive systems, and specifically when used on cloud environments, because of its complexity and dynamic nature. The main goal of this thesis project is to investigate different methods for the specification of adaptive systems, and to propose techniques and tools for the modeling of self-adaptive systems and environments, considering adaptation mechanisms, and approaches for the estimation of different Quality of Service (QoS) metrics that help in the analysis of the systems to be developed. Specifically, we will provide generic mechanisms for the modelling of adaptive systems and environments, the definition of metrics as transformation rules, and tools using such system specifications for their analysis. We will focus on the kind of systems and adaptation mechanisms we find in the cloud, and will evaluate our proposal on state-of-the-art cloud applications. We will present an approach of predictive analysis, based on graph transformation, which provides the capability of taking decisions about elasticity-related QoS from the definition of an adaptive model of the system and the specification of adaptation mechanisms described in a formal language to control elasticity in cloud applications. The proposed approach will enable the simulation of cloud environments and their elasticity features at design time, which allows the prediction of different QoS metrics in cloud scenarios, and provides the capability of specifying and tuning elasticity monitoring, constraints, and strategies at different levels.

1 Research Problem and Motivation

An important feature of self-adaptive systems, such as cloud applications, is the capability to react to changes dynamically, which brings new challenges from the perspective of predictive analysis. Elasticity is one of the key issues in Cloud Computing, where in fact, elasticity refers not only to resources but also to quality and cost, and how they can influence each other when changing the context. Therefore, having the capability of predicting problems related with performance

constraints, scalability issues or reliability risks, becomes particularly relevant when adaptive systems are being modelled. Predictive information related with Quality of Service (QoS) metrics allows the adoption of decisions before the deployment phase, so preventing problems and mitigating the impact of bad design or management decisions. In order to get this kind of useful information, we need to have precise and appropriate models both of the application to be analysed and of the context (cloud platform or self-adaptive system) where the application is going to be deployed. The more accurate the model is, the more precise the prediction. In addition to the possibility of getting predictive information, the same scenario may also help in the calibration of quality parameters, providing support to estimate optimal values for them.

A number of different predictive analysis approaches can be found in the literature, including techniques based on stochastic networks, Petri nets, statistical methods or simulation. However, current predictive analysis tools present limitations to deal with dynamic architectures. The main limitations exhibited by most current tools comes from the fact that they do not allow changes in the model along the analysis process, operating just on static structures. This is the case, for instance, of the Palladio tool [1], one of the currently more successful predictive analysis frameworks, and widely used both in industry and academia. Thus, for instance, the Palladio Simulator can be used to predict QoS properties (performance and reliability) from software architecture models.

Although Palladio group is introducing the cloud and adaptive systems issues in some recent work [2, 3] the limitation of using static architectures has not still been solved. There have been other attempts to deal with these issues, such as those in [4] to build support for dynamic systems on Palladio. Other tools such as D-Klaper [5], MEDEA [6] or the one described in [7] have proposed alternative solutions for performance prediction of dynamic systems. However, much work needs still to be made to solve the problem in a convenient way as it is explained in Sect. 3.

2 Research Challenges

With the aim of extending the Palladio's predictive capabilities to support dynamic systems, we propose to use the e-Motions implementation of the Palladio Component Model (PCM) described in [8]. Given the metamodel of Palladio and its operational semantics expressed in terms of graph-transformation rules, this implementation allows to analyse (static) systems modeled in Palladio Bench. However, the relevance of this e-Motions specification is the capability of integrating new adaptation rules, which extend the behaviour of Palladio, making the analysis of dynamic systems feasible, and increasing its expressiveness [9].

In order to explain the contribution intended of this thesis project, the diagram in Fig. 1 summarizes the relationship between the main processes and elements of the proposal. In e-Motions, a DSL is specified by its syntax (a metamodel) and a behavior (an operational semantics described as a set of graph

transformation rules). The systems to be analyzed are specified using Palladio, and specifically the Palladio Component Model (PCM). Models conforming to the PCM are composed of four different submodels, which correspond to different and complementary views of the system. The Palladio language is specified in e-Motions by taking an extended PCM, denoted PCM^* , which includes definitions for tokens and dynamics of systems, and its behavior. As presented in [8], static systems defined in the Palladio Bench (models M_{app} conforming to the PCM) can be loaded and analyzed in e-Motions using the DSL $\text{PCM}^* + \text{Beh}_{\text{Palladio}}$.

To deal with adaptive systems, the behavior to be used is extended with adaptation mechanisms, specified as additional e-Motions transformation rules, $\text{Beh}_{\text{Adaptation}}$. To control the application of the adaptation operations, SYBL (Simple Yet Beautiful Language) [10] is used, which is a language for controlling elasticity in cloud applications in runtime. For a specific model M_{app} , the SYBL annotations provided are represented as a model extension $\text{Ctrl}_{\text{SYBL}}$ so that the extended model $M_{\text{app}} + \text{Ctrl}_{\text{SYBL}}$ can be used to analyze the performance of the described adaptive system using the DSL $\text{PCM}^* + (\text{Beh}_{\text{Palladio}} + \text{Beh}_{\text{Adaptation}})$.

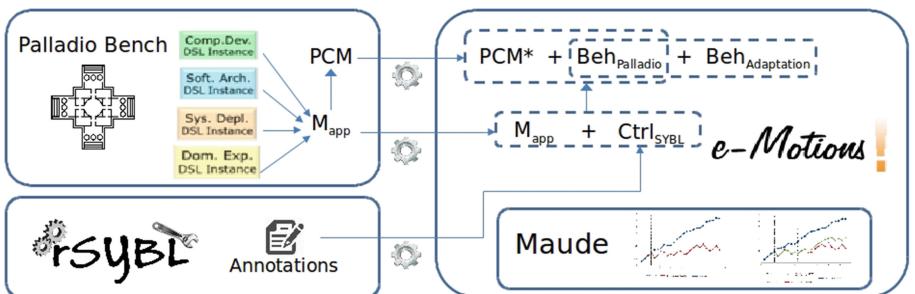


Fig. 1. Architecture for the specification of cloud systems

Our proposal will try to offer the techniques and tools to allow the modeling of self-adaptive systems, and specifically in cloud infrastructures, and their analysis, so that a better estimation of the satisfaction of the requirements of systems can be carried on, supporting a better selection of resources and a better calibration of the operational parameters.

Our claim is that by extending the capacity of the Palladio system, and specifically its PCM and tooling, will allow us to perform more precise statistical analysis of self-adaptive systems, including those reconfiguration algorithms usually found in cloud environments.

3 Related Work

In this section, we discuss other approaches that use predictive strategies, most of them for performance prediction at runtime or at design time.

Huber et al. propose in [4] a DSL to describe the behaviour of self-adaptive systems based on strategies, tactics and actions. This work is part of the Descartes project, which uses Palladio PCM for their design time phases. However, they focus on runtime performance analysis, not in predictive analysis of applications at design time.

SimuLizar [3] is an extension of Palladio for the performance analysis of self-adaptive systems at design time. However, the simulation scope is limited to only a set of rules that are triggered between the static environment models, which prevents from testing all possible reachable states of the systems.

D-Klaper [5] is a tool for model-driven performance engineering which can be applied to self-adaptive systems. It uses an intermediate language to provide software design models, which can then be analyzed. However, the D-Klaper language does not support the modeling of adaptation rules, nor the transformation of input models.

MEDEA [6] is an approach that proposes the performance prediction at the beginning of its life cycle for this, modeled the workloads with the resource consumption, capturing the CPU, memory and disks. However, although using models to represent the system, this is used to generate executable code for real hardware and middleware deployments and the results of the executions are presented to the expert through specific context views that indicate whether the design meets the performance requirements, not acting entirely at design time.

Johnsen et al. present in [7] an approach model-based prediction to compare the effect, in terms of performance and accumulated cost, of selecting different instance types for virtual servers from Amazon Web Services (AWS), for this their used a highly configurable modeling framework for applications running on Apache YARN, the ABS-YARN, which using the executable semantics of Real-Time ABS, defined in Maude, as a simulation tool. However, they do not model the application but use values obtained from real measurements.

CloudScale method [2] aims to analyze and identify the scalability problem in design time. This approach presents the ScaleDL languages, which consist in the set of languages required to allow software architects managing scalability, elasticity, and cost-efficiency. This set of languages includes the technologies mentioned above Simulizar and Descartes. Although it is a robust method, the Palladio limitations — as no possibility of modifications of the initial models — remain present in the approach.

4 Proposed Solution and Preliminary Results

Building on the already available e-Motions specification of Palladio, we have already specified several adaptation mechanisms by providing appropriate adaptation rules as graph transformation rules. Specifically, we can specify a system in Palladio, including descriptions of its components architecture, environment and usage model, and perform its QoS analysis in e-Motions using transformation rules that describe the adaptation mechanisms available in the system and its environment.

We were able to operate simulations and perform predictive analysis taking into account some the adaptation mechanisms, and thus showing the feasibility of the approach. So far, we have been able to model the dynamic adaptation of the system in accordance to its continuous monitoring by creating rules for: load balancing, variable usage, the scale in/out nodes and the scale up/down resources capacity (specifically CPU).

As early experiments, presented in [9] and [11], we were able to perform simulation-based predictive analysis of adaptive systems. Our approach is able to operate simulations and perform predictive analysis taking into account different adaptation mechanisms. For the case study used we observed the response time, throughput at resource usage of the system in order to show the feasibility of the approach.

The prediction of QoS metrics is one of the most relevant issues when gathering knowledge of applications and their environments, compared to other solutions presented in the literature [12]. However, existing prediction methods do not consider specific cloud metrics and, therefore, they are not capable of managing other particular cloud features, such as self-provisioning on demand, measured usage, network access, resource pooling, and elasticity. Properties related to scalability, elasticity and efficiency are essential to achieve a dynamic adaptation in a cloud scenario, specifically for resource allocation and pay-per-use. Thus, we need to take into account these new metrics [13], and also a taxonomy of different sources of uncertainty present in the models of self-adaptive systems and the different ways of managing them [14].

5 Plan for Evaluation and Validation

Building on the knowledge we have gathered so far, we will specify other mechanisms of adaptation available for cloud systems in more ambitious case studies. We will consider other QoS metrics, including not only performance metrics, but also others related to feasibility, costs, and security.

With this thesis, we intend to offer the techniques and tools to allow the modeling of self-adaptive systems, and specifically cloud infrastructure, and their analysis, so that a better estimation of the satisfaction of the requirements of systems can be carried on, supporting a better selection of resources and a better calibration of the operational parameters.

To perform the analysis of a dynamic system, it is necessary to consider their capacity to process and manage different workflows, react through variations of the usage, and to carry on the necessary changes when components have assigned different workloads. Following standard techniques, we will model workloads based on real uses of systems, and will use this information to perform our simulations.

We will evaluate our proposal modeling real applications running in real cloud environments, and will verify that the results produced by our predictive analysis match the actual behavior of the real system executed in the cloud. We will use standard techniques, such as the Kolmogorov-Smirnov test [15] and other similar ones, for the comparison of the probability distributions.

References

1. Happe, J., Koziolek, H., Reussner, R.: Facilitating performance predictions using software components. *IEEE Softw.* **28**(3), 27–33 (2011)
2. Becker, S., Brataas, G., Lehrig, S.: Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications: The CloudScale Method. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-319-54286-7>
3. Becker, M., Becker, S., Meyer, J.: Simulizar: design-time modeling and performance analysis of self-adaptive systems. *Softw. Eng.* **21***3*, 71–84 (2013)
4. Huber, N., van Hoorn, A., Koziolek, A., Brosig, F., Kounev, S.: S/T/A: meta-modeling run-time adaptation in component-based system architectures. In: 2012 IEEE Ninth International Conference on e-Business Engineering (ICEBE), pp. 70–77 IEEE (2012)
5. Grassi, V., Mirandola, R., Randazzo, E.: Model-driven assessment of QoS-aware self-adaptation. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 201–222. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02161-9_11
6. Falkner, K., Szabo, C., Chiprianov, V.: Model-driven performance prediction of systems of systems. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, p. 44. ACM (2016)
7. Johnsen, E.B., Lin, J.-C., Yu, I.C.: Comparing AWS deployments using model-based predictions. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9953, pp. 482–496. Springer, Cham (2016)
8. Moreno-Delgado, A., Durán, F., Zschaler, S., Troya, J.: Modular DSLs for flexible analysis: an e-motions reimplementation of palladio. In: Cabot, J., Rubin, J. (eds.) ECMFA 2014. LNCS, vol. 8569, pp. 132–147. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09195-2_9
9. de Oliveira, P.A., Moreno-Delgado, A., Durán, F., Pimentel, E.: Towards the predictive analysis of cloud systems with e-Motions. In: XX Ibero-American Conference on Software Engineering (CIbSE) (2017)
10. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: SYBL: an extensible language for controlling elasticity in cloud applications. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 112–119. IEEE (2013)
11. de Oliveira, P.A., Durán, F., Pimentel, E.: Graph-transformation for the performance analysis of elastic systems. In: 15th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems (2017)
12. Chinneck, J., Litoiu, M., Woodside, M.: Real-time multi-cloud management needs application awareness. In: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering, ICPE 2014, New York, NY, USA, pp. 293–296. ACM (2014)
13. Becker, M., Lehrig, S., Becker, S.: Systematically deriving quality metrics for cloud computing systems. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, pp. 169–174. ACM (2015)
14. Perez-Palacin, D., Mirandola, R.: Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering, ICPE 2014, New York, NY, USA, pp. 3–14. ACM (2014)
15. Hollander, A., Wolfe, D.: Nonparametric Statistical Methods. Wiley-Interscience, New York (1999)



Paving the Way for a Real-Time Context-Aware Predictive Architecture

David Corral-Plaza¹✉, Guadalupe Ortiz², and Juan Boubeta-Puig²

¹ School of Engineering, University of Cádiz,
Avda. de la Universidad de Cádiz 10, Puerto Real, 11519 Cádiz, Spain
david.corral@uca.es

² Department of Computer Science and Engineering, University of Cádiz,
Avda. de la Universidad de Cádiz 10, Puerto Real, 11519 Cádiz, Spain
{guadalupe.ortiz, juan.boubeta}@uca.es

Abstract. Internet of Things society generates and needs to consume huge amounts of data in a demanding context-aware scenario. Such exponentially growing data sources require the use of novel processing methodologies, technologies and tools to facilitate data processing in order to detect and prevent situations of interest for the users in their particular context. To solve this issue, we propose an architecture which making use of emerging technologies and cloud platforms can process huge amounts of heterogeneous data and promptly alert users of relevant situations for a particular domain according to their context. Last, but not least, we will provide a graphical tool for domain experts to easily model, automatically generate code and deploy the situations to be detected and the actions to be taken in consequence. The proposal will be evaluated through a real case study related to air quality monitoring and lung diseases in collaboration with a doctor specialist on lung diseases of a public hospital.

Keywords: Prediction · Context-awareness · Internet of Things
Event-Driven Service-Oriented Architectures

1 Motivation and Problem Statement

Nowadays, society has numerous methods for consumption, production and exchange of huge amounts of data. This data is mainly originated in the multitude of existent devices and channels: social networks, smart devices such as smartphones, tablets or wearables, Internet of Things (IoT) devices, et cetera. Such exponentially growing data sources require the use of processing methodologies, and tools different from conventional ones in order to fast profit from the relevant information obtained. Even more, in the smart world [1] context awareness plays a highly relevant role which must be taken into account when providing the information to the final recipient. Dey et al.'s context definition in [2] is specially well-known: “*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*”.

Therefore, it is vital to be able to process such large amounts of data together with users' context [3], in order to detect real-time situations of interests that are relevant for a particular user, context and domain. Moreover, we have to follow real-time analysis not only to detect situations of interest, but also to predict them so that we can take necessary actions in advance to prevent user from undesirable situations.

The problem is that, currently, an efficient integration of all the stages implied in this process is not provided. We can find systems to process big amounts of data in real time [4, 5], systems which are aware of the user's context [6], and systems which apply prediction algorithms to vast amounts of data [7]. However, when trying to integrate all of them, we have several problems: (1) the *variety* of data formats required to be managed by an unique system, given the heterogeneity of the IoT sensing data; (2) the handicap of *predicting* situations of interest from such heterogeneous data, which is related to different domains and contexts; (3) the *volume* and *velocity* of this data, becoming a problem to achieve system scalability and performance gains; and (4) the *domain experts* that do not have enough technical knowledge to define what they need to detect or predict (the *data value*) with current tools. These problems lead us to the research challenges explained in the following section.

2 Research Challenges

We will deal with the following research challenges:

1. First of all, data will have to be gathered and processed in real time. As discussed before, the source where these data come from will be very varied—for example, social networks, smartphone sensors, IoT sensors, et cetera—; therefore, we will need a means to homogenize all these data with an appropriate structure to be used in the data analysis systems and an easy procedure to add new sources. The real challenge will be to define a unified model useful for a wide range of domains.
2. Besides, we need a system which able to constantly detect the user context and to provide a consistent response according to real-time circumstances of a given user.
3. Not only that, a collaborative architecture is required so that several parties can provide context data and relevant events to enrich the relevant available data.
4. Suitable novel machine-learning methodologies and tools to process and correlate huge amounts of data in order to detect and predict situations of interest in real time for a particular user in a particular domain. It is essential that the architecture is efficient and scalable [8].
5. Finally, domain experts should be able to easily design the situations to be detected and predicted in real time, as well as the actions to be taken consequently, depending on a particular set of incoming data and context information.

3 Proposed Solution

We envision a holistic Event-Driven Service-Oriented Architecture (ED-SOA), assuming that anything that happens is an event. Such envisioned architecture, shown in Fig. 1, should contain the following modules:

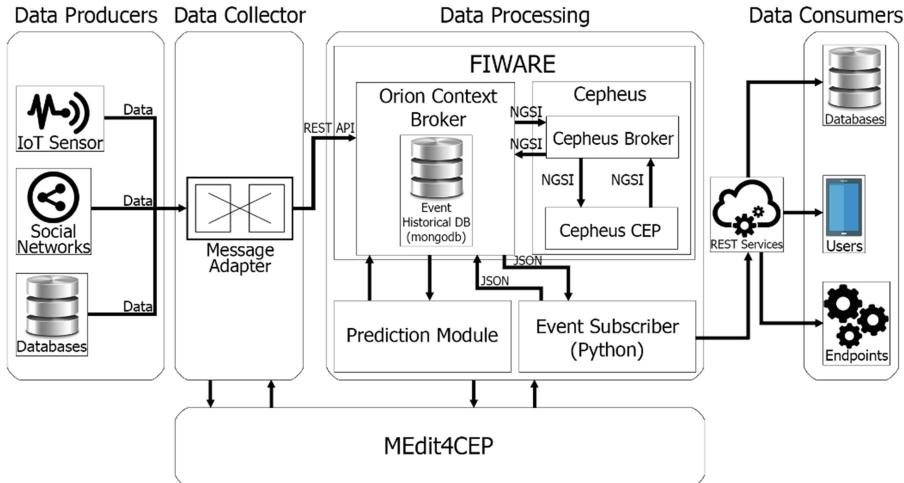


Fig. 1. Proposed architecture

- Firstly, *Data producers* should gather data from several sources (databases, IoT sensors, social networks, et cetera) and send them to the data collector.
- Secondly, *Data Collector* follows the necessary transformations so that the information received can be used in the following phases of our solution. It is an intermediate layer that performs a process of homogenization since information will most probably be received in different formats and structures in most scenarios.
- Thirdly, *Data Processing* should provide Complex Event Processing (CEP), context-awareness and prediction module. Initially we bet for FIWARE [9], a Platform as a Service (PaaS) development tool in the cloud.
- Fourthly, we have *Data Consumers*, which can be databases, end users or additional endpoints which pave the way for the collaborative architecture. Such data consumers communicate with the previous module through a REST interface, however additional protocols might be required.
- Finally, in the bottom of Fig. 1, we can see the graphical modeling tool for pattern and actions definition for FIWARE. Such tool is expected to be an extension of MEdit4CEP [10], with the goal of domain experts to easily model events and patterns expected in a their specific application domain and being able to deploy them, automatically, in the processing system that is currently running in FIWARE with no need of programming knowledge.

4 Preliminary Results

By the time being, we have implemented and tested part of the proposed architecture:

1. The data producers integrated are databases, IoT sensors, IoT platforms and message queues. Integrating social networks is our following step to be done here.
2. Currently, data collector is being improved; we initially implemented such data collector and homogenization process through an Enterprise Service Bus (ESB); however our experience, supported also with other results in the research group [3, 11], shows that the ESB decreases the system performance.
3. Among the different enablers existing in the FIWARE catalog, we already started using Orion Context Broker and Cepheus: Orion has become the brain of our application and control all the information that the different modules receive and emit inside FIWARE. Cepheus is a CEP engine which will let us define the event patterns to be monitored. We did not start yet working on the prediction module.
4. In the fourth stage we included databases and mobile applications and we are working on the inclusion of message queues to foster the collaboration [11].
5. The work on the domain experts graphical tool is not yet started.

In order to test the proposed architecture, we particularized it for air quality domain. We have analyzed and reported air observations made around the Andalusian territory for several months. For this purpose we have connected the data provided from Andalusian air quality stations as the main data source for our system, we have created an Android app which monitors the user context and we have provided personalized alerts according to the sensed air quality and user context. We also measured the performance of the system and even though the system was clearly efficient for the case study in question. As a result, we detected that our initial implementation was not scalable enough, reason why we are working now on improving it.

5 Related Work

Concerning prediction architectures where SOA and CEP are used, Mousheimish et al. [7] propose one which is able to generate CEP rules for any person. The authors extend their work in [12], for an artworks transportation case study and use an additional framework to monitor relevant values and predict if they are going to take undesired values according to a previously specified Service Level Agreement. The main limitation of their proposals is that they do not take into account several inputs sources.

Concerning proposals based on FIWARE, Fazio et al. [6] make use of FIWARE to design a real e-health remote patient monitoring architecture, which allow caregivers to improve remote assistance to patients at home. Wolfert et al. [4] describe how FIWARE has been applied to the smart farming domain, given support to tackle the data chain of big data applications: data capture, data storage, data transfer, data transformation, data analytics and data marketing. Fernández et al. [5] have developed SmartPort, a FIWARE-based platform for sensor data monitoring in a seaport located in Gran Canaria, Spain. Comparing these proposals with our proposed work, most of them do not benefit from using the CEP technology and Cepheus component to

automatically detect meaningful patterns in real time; Fazio et al.'s one is an exception. In addition, they do not provide prediction mechanisms.

6 Evaluation Plan

As could not be otherwise, during the development of the PhD, we will continuously review the literature related to the problem stated and the proposed solutions.

On the other hand, we will carry out different experiments in different application domains to evaluate in an empirical way the performance of the proposed architecture and solution. Among others, we will go on with the air quality scenario.

In this same line, we expect to test a system to detect air quality alerts with people suffering from all types of lung diseases in collaboration with Dr. Carmen Maza Ortega, a specialist in lung diseases at the *Hospital Universitario de Puerto Real* (Spain).

7 Planned Timeline

In this section we schedule the tasks to be performed during the thirty-six months expected for the development of the PhD, which we roughly identify between November 2017 and November 2020 (see Fig. 2).

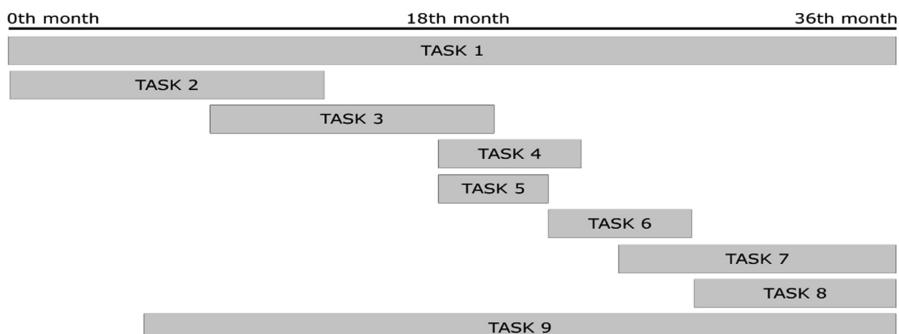


Fig. 2. Estimated timeline for the PhD

- Task 1. Reviewing literature related to problem statement and proposed solution.
- Task 2. Studying emerging and well-established technologies and tools for data processing with the aim of detecting and predicting particular situations of interest.
- Task 3. Developing the proposed architecture according to previous tasks.
- Task 4. Evaluating the architecture through a case study in a relevant domain.
- Task 5. Facilitating the addition of new sources and formats.
- Task 6. Incorporating MEdit4CEP or an alternative tool in the architecture to permit pattern and action graphical design and code generation and deployment.
- Task 7. Evaluating the proposed architecture thorough a real case study.
- Task 8. Writing and defending the PhD.
- Task 9. Disseminating the research results in conferences and journals.

8 Conclusions

We have outlined a starting PhD focused on providing real-time context-aware detection and prediction under the smart world demanding scenario. We envision a solution based on an ED-SOA in the cloud combining several cutting-edge technologies which will provide us with early context-aware notifications and predictions in a particular domain of application. The system envisioned is designed to be scalable and highly configurable, so that it can integrate an undefined and heterogeneous number of data sources. Moreover, the system will be easily applicable to several domains, and domain experts will be provided with a graphical modeling tool which will prevent them from coding and configuration issues, facilitating the architecture widespread use.

References

1. Liu, H., Ning, H., Mu, Q., Zheng, Y., Zeng, J., Yang, L.T., Huang, R., Ma, J.: A review of the smart world. *Future Gener. Comput. Syst.* **56**, 684–700 (2017)
2. Dey, A.K.: Understanding and using context. *Pers. Ubiquit. Comput.* **5**, 4–7 (2001)
3. de Garcia Prado, A., Ortiz, G., Boubeta-Puig, J.: CARED-SOA: a context-AwaRe event-driven service oriented architecture. *IEEE Access* **5**, 1–18 (2017)
4. Wolfert, S., Ge, L., Verdouw, C., Bogaardt, M.-J.: Big data in smart farming – a review. *Agric. Syst.* **153**, 69–80 (2017)
5. Fernández, P., Santana, J.M., Ortega, S., Trujillo, A., Suárez, J.P., Domínguez, C., Santana, J., Sánchez, A.: SmartPort: a platform for sensor data monitoring in a seaport based on FIWARE. *Sensors* **16**, 417 (2016)
6. Fazio, M., Celesti, A., Márquez, F.G., Glikson, A., Villari, M.: Exploiting the FIWARE cloud platform to develop a remote patient monitoring system. In: 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 264–270 (2015)
7. Mousheimish, R., Taher, Y., Zeitouni, K.: autoCEP: automatic learning of predictive rules for complex event processing. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 586–593. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46295-0_38
8. Papazoglou, M.: Web Services and SOA: Principles and Technology. Pearson Education, New York (2012)
9. Core Platform of the Future Internet: FIWARE. <http://www.fiware.org/>
10. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: MEdit4CEP: a model-driven solution for real-time decision making in SOA 2.0. *Knowl. Based Syst.* **89**, 97–112 (2015)
11. Garcia-de-Prado, A., Ortiz, G., Boubeta-Puig, J.: COLLECT: COLlaborativE ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things. *Expert Syst. Appl.* **85**, 231–248 (2017)
12. Mousheimish, R., Taher, Y., Zeitouni, K., Dubus, M.: PACT-ART: enrichment, data mining, and complex event processing in the internet of cultural things. In: 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 476–483. IEEE (2016)

Demonstration

Introduction to the Demonstration Track

Preface

The ICSOC 2017 Demonstration Track was held in conjunction with the 15th International Conference on Service Oriented Computing (ICSO 2017) on 13–16 of November in Málaga, Spain. This track offered an exciting and highly interactive opportunity to show research prototypes in service oriented computing (SOC) and related areas. These research prototype demos focused on developments and innovation in the areas of service engineering, operations, cloud and big data services, implementation of services as well as development and adoption of services in specific organizations, businesses, and the society at large.

We received 12 submissions and accepted 6. These demos clearly showed interesting improvements and significance from recently implemented systems, and offered possibilities for fruitful discussions.

- Saatkamp, Karoline; Breitenbücher, Uwe; Képes, Kálmán; Leymann, Frank; Zimmermann, Michael. *OpenTOSCA Injector: Vertical and Horizontal Topology Model Injection*
- Gschwind, Katharina; Adam, Constantin; Duri, Sastry; Nadgowda, Shripad; Vukovic, Maja. *Optimizing Service Delivery with Minimal Runtimes*
- Wolters, Dennis; Heindorf, Stefan; Kirchhoff, Jonas; Engels, Gregor. *Semantic Data Mediator: Linking Services to Websites*
- Sandobalin, Julio; Insfran, Emilio; Abrahao, Silvia. *ARGON: A Tool for Modeling Cloud Resources*
- Mohamed, Mohamed; Engel, Robert; Warke, Amit; Ludwig, Heiko. *Ubiquity: An Extensible Framework for Persistence in Container Environments*
- Khochare, Aakash; Ravindra, Pushkara; Reddy, Siva Prakash; Simmhan, Yogesh. *Distributed Video Analytics across Edge and Cloud using ECHO*

We would like to thank the authors for their submissions, the program committee for their reviewing work, and the organizers of the ICSOC 2017 conference for their support which made this demo track possible.

Nima Kaviani
Manuel Lama Penin

Organization

Demonstration Chairs

Nima Kaviani
Manuel Lama Penin

IBM Cloud Labs, San Jose, USA
University of Santiago de Compostela, Spain

Program Committee

Pedro Álvarez	University of Zaragoza, Spain
Mohsen Asadi	SAP, CA, Canada
Djamal Benslimane	University of Lyon, France
Athman Bouguettaya	The University of Sydney, Australia
Ivona Brandic	Vienna University of Technology, Austria
Antoni Brogi	University of Pisa, Italy
Francois Charoy	University of Lorraine, France
Jürgen Cito	University of Zurich, Switzerland
Florian Daniel	Politecnico di Milano, Italy
Zhiyong Feng	Tianjin University, China
Marios-Eleftherios Fokaefs	York University, Canada
Nam Giang	UBC, Canada
Armin Haller	Australian National University, Australia
Raman Kazhamiakin	Fondazione Bruno Kessler, Italy
Philippe Lalanda	Joseph Fourier University, France
Philipp Leitner	University of Zurich, Switzerland
Xumin Liu	Rochester Institute of Technology, USA
Bardia Mohabbati	Amazon, USA
Mohamed Mohamed	IBM Almaden Research Center, USA
Guadalupe Ortiz	University of Cádiz, Spain
Helen Paik	UNSW, Australia
Barba Pernici	Politecnico di Milano, Italy
Pierluigi Plebani	Politecnico di Milano, Italy
Mohammad Sadoghi	Purdue University, USA
Andreas Solti	Institute for Information Business, Austria
Juan C. Vidal	University of Santiago de Compostela, Spain
Zhongjie Wang	Harbin Institute of Technology, China
Barbara Weber	Technical University of Denmark
Jianwei Yin	Zhejiang University, China



OpenTOSCA Injector: Vertical and Horizontal Topology Model Injection

Karoline Saatkamp^(✉), Uwe Breitenbücher, Kálmán Képes, Frank Leymann,
and Michael Zimmermann

Institute of Architecture of Application Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany

{Saatkamp,Breitenbucher,Kepes,Leymann,Zimmermann}@iaas.uni-stuttgart.de

Abstract. The automation of application deployments is supported by various technologies. The TOSCA standard facilitates to describe application deployments in a portable manner by modeling application structures as topology models. The final structure often depends on the target environment and is, therefore, not always known at modeling time. However, a manual adaptation is error-prone and time-consuming. In this paper, we demonstrate the OpenTOSCA Injector for an automated completion of topology models: the extended TOSCA runtime OpenTOSCA for an automated injection and deployment is presented.

Keywords: TOSCA · Deployment model · Completion automation

1 Introduction and Motivation

In recent years, several technologies and standards were developed to automate the deployment of cloud applications. This includes configuration management technologies such as Chef, container technologies such as Docker, and standards such as the Topology and Orchestration Specification for Cloud Applications (TOSCA) [6]. TOSCA is an OASIS standard that enables to define application deployments by topology models and management plans, which can be executed automatically by a TOSCA runtime, e.g., the OpenTOSCA container [1].

A topology model describes the application components and their relations. This includes application-specific components, such as PHP applications or databases, middleware, and infrastructure components, such as web servers or virtual machines. Thereby, application deployments can be described in a vendor-independent and portable manner. However, the available middleware, infrastructure, as well as application-specific components can differ between environments. When, for example, application deployments are provided for third parties or parts of the IT infrastructure are outsourced, the target environment is not known in advance. Thus, the final topology model is not known at modeling time. However, the manual adaptation for each target environment is time-consuming and error-prone [4]. To enable an environment-independent modeling

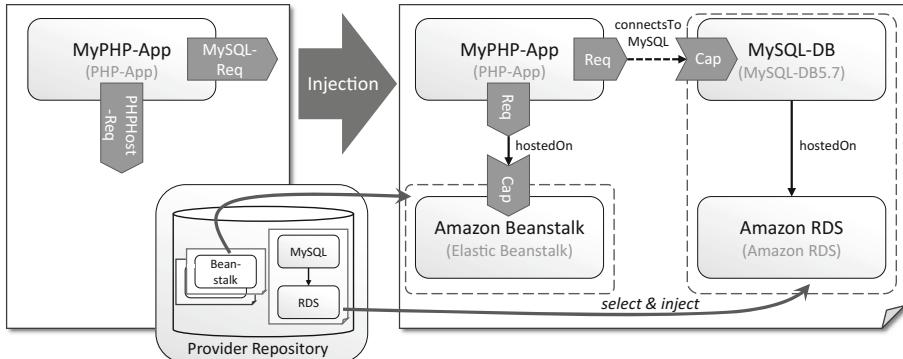


Fig. 1. Topology model with open requirements (left) and injected components (right)

via incomplete topology models and an automated environment-specific injection of components during deployment time, the *OpenTOSCA Injector* is developed: infrastructure components (vertical injection) as well as, e.g., data storage stacks (horizontal injection) are selected and injected to complete formerly incomplete topology models.

2 TOSCA Fundamentals and Injection Concept

As already mentioned TOSCA is an OASIS standard that enables to describe the automated deployment of applications in a vendor-independent and portable manner. Several TOSCA runtimes to process TOSCA models are already developed such as Cloudify¹, Apache ARIA TOSCA², and the OpenTOSCA container³. In the following all TOSCA concepts relevant for the OpenTOSCA Injector are introduced. More details about TOSCA can be found in the specification [6].

The structure of an application can be described as *Topology Template*, which is a directed and weighted multigraph as depicted in Fig. 1 on the right. The components are modeled as *Node Templates*, e.g., MyPHP-App, and the relations between them as *Relationship Templates* such as *hostedOn*. Their semantic is defined by *Node Types*, e.g., PHP-App, and *Relationship Types*, respectively. Types can be derived from other types, thus, inheritance hierarchies can be defined. For Relationship Types valid target and source elements are specified, which can be Node Types or *Requirement Types* and *Capability Types*. For each Requirement Type, exactly one *requiredCapabilityType* is defined, i.e., each Capability of this type can be matched to a Requirement of the respective Requirement Type. *Requirements* and *Capabilities* of these types can be attached to Node Templates. Thus, the matching between Requirements and

¹ <http://cloudify.co/>.

² <http://ariatosca.incubator.apache.org/>.

³ <http://www.opentosca.org/>.

Capabilities and hence between Node Templates is realized, which is the basis for the OpenTOSCA Injector.

The left side of Fig. 1 shows an incomplete topology with two open Requirements. The Requirements *PHPHost-Req* and *MySQL-Req* require Node Templates with matching Capabilities. Possible suitable Node Templates are stored in a local *Provider Repository* in which the respective owner can add all available components in the environment, such as specific infrastructure components. However, in future work also the linkage to public repositories should be enabled. With the OpenTOSCA Injector, not only single Node Templates but also topology fragments can be injected [4]. As shown in Fig. 1 on the right, a topology fragment could consist of a *MySQL-DB* and an *Amazon RDS* component, which is injected based on the matching between the Requirement *MySQL-Req* and the specified requiredCapabilityType, e.g., *MySQL-Cap* attached to the *MySQL-DB*.

For each match a suitable Relationship Type has to be found to connect the matched Node Templates. This, for example, could be a *hostedOn* or *connectsTo* relation. The suitable Relationship Type is determined by the assigned Requirement and Capability. However, specific types such as *connectsToMySQL* are not always available in the target environment. For this, TOSCA base types are used: the *hostedOn* and the *connectsTo* Relationship Type [7]. In any case, one of these base types is selected, because of the predefined inheritance hierarchy of Capability Types. After the Node Templates or topology fragments are injected with suitable Relationship Types, the topology is complete and deployable. We implemented the described injection concept and demonstrate it with the OpenTOSCA Injector, which extends the existing OpenTOSCA container.

3 System Architecture and Demonstration

The OpenTOSCA container is a TOSCA runtime supporting the imperative and declarative processing of TOSCA models for an automated deployment [2]. For the imperative processing the management plans are explicitly defined, whereas at the declarative processing the deployment logic is inferred from the topology model. Our demonstration is based on declarative provisioning modeling and

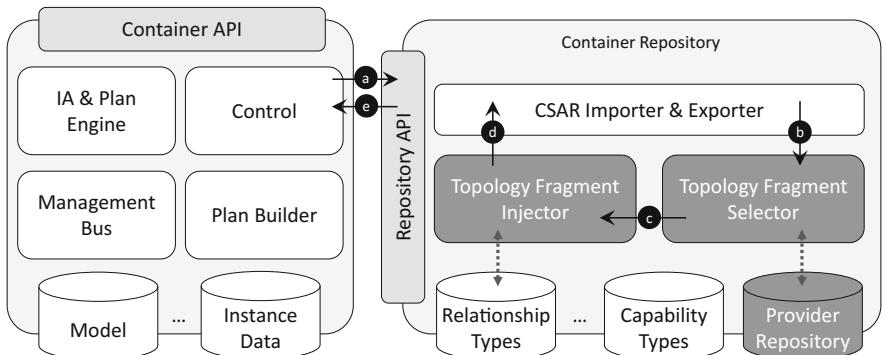


Fig. 2. Extended OpenTOSCA system architecture and processing overview

management plans are not explicitly considered. Besides the actual purpose of the OpenTOSCA container to deploy cloud application, it is also used for the automated deployment of use cases of the 4th Industrial Revolution [3] and IoT scenarios with different messaging middleware systems [9, 10]. In this demonstration the OpenTOSCA container is used to automatically complete and deploy topology models for different deployment environments.

In Fig. 2 the extended OpenTOSCA system architecture⁴ is depicted. While the left hand side shows the existing OpenTOSCA components required for the deployment, the right hand side shows the *Container Repository* extending the existing runtime. The Container API is used to upload topology models and all related artifacts, such as JAR files or scripts for the deployment. The *Control* component is responsible for interpreting the topology and tracking the process. For a declarative processing, the *Plan Builder* generates plans based on the topology model. The operations invoked by the plans need Implementation Artifacts (IA) to install and start the application's components. They are part of the upload and processed by the *IA Engine*, while the plans are processed by the *Plan Engine*. With the *Management Bus*, plans finally invoke different kinds of management operations for the deployment. All data required during the deployment, e.g., model information, and after the deployment such as the instance data, are stored in databases.

For the demonstration of the injection the *Container Repository* is essential. Its source code is based on the *Eclipse Winery*⁵, a modeling tool for TOSCA [5]. Because the injection affects the topology model, the existing Winery capabilities to deal with topology model elements is utilized. The *Topology Fragment Injector*, the *Topology Fragment Selector* component, and the *Provider Repository* extend the existing Winery source code to use it as Container Repository for the injection. For the injection, the incomplete topology as depicted in Fig. 1 on the left is uploaded to the Container API and forwarded to the Control component. It checks the topology model for open requirements and in case open requirements are contained, an injection request is sent to the Container Repository (cf. (a) in Fig. 2). The Topology Fragment Selector browses the Provider Repository for topology fragments with matching Capabilities (cf. (b) in Fig. 2). For multiple injection options, the user can select the preferred fragment. After the selection, suitable Relationship Templates are determined based on the Requirements and Capabilities, and used to inject the topology fragments in the model (cf. (c) in Fig. 2). The completed topology model as presented in Fig. 1 is exported and the Control component starts the deployment of the application (cf. (d) in Fig. 2).

The demonstrated OpenTOSCA Injector implements the TOSCA concept for Requirement and Capability matching in an automated manner. It facilitates, beyond the general matching of Capabilities, the injection of whole topology fragments. The objective is to model an incomplete topology model with defined requirements which is completed depending on the specific deployment environment, e.g., a factory, company, or public cloud provider. The Injector can be used

⁴ <https://github.com/OpenTOSCA>.

⁵ <https://github.com/eclipse/winery>.

for the completion by, e.g., different infrastructure components (vertical injection) such as an OpenStack or vSphere depending on the available infrastructure as well as for the connection with different data sources for example to analyze the available data in an environment (horizontal injection). Additionally, it supports to restrict the set of considered topology fragments for the injection by target labels attached to Node Templates to express preferences for the matching [8]. With the OpenTOSCA Injector, concepts for an environment-dependent and automated application deployment can be realized.

Acknowledgments. This work was partially funded by the projects SePiA.Pro (01MD16013F), SmartOrchestra (01MD16001F), and IC4F (01MA17008G).

References

1. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA – a runtime for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 692–695. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_62
2. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining declarative and imperative cloud application provisioning based on TOSCA. In: International Conference on Cloud Engineering, pp. 87–96. IEEE (2014)
3. Falkenthal, M., Breitenbücher, U., Képes, K., Leymann, F., Zimmermann, M., Christ, M., Neuffer, J., Braun, N., Kempa-Liehr, A.W.: OpenTOSCA for the 4th industrial revolution: automating the provisioning of analytics tools based on apache flink. In: Proceedings of the 6th International Conference on the Internet of Things, pp. 179–180. ACM (2016)
4. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F., et al.: Automatic topology completion of TOSCA-based cloud applications. In: GI-Jahrestagung, GI, vol. P-251, pp. 247–258. GI (2014)
5. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – a modeling tool for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 700–704. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_64
6. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0. OASIS (2013)
7. OASIS: TOSCA Simple Profile in YAML Version 1.0. OASIS (2015)
8. Saatkamp, K., Breitenbücher, U., Kopp, O., Leymann, F.: Topology splitting and matching for multi-cloud deployments. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science, pp. 247–258. SciTePress (2017)
9. Franco da Silva, A.C., Breitenbücher, U., Hirmer, P., Képes, K., Kopp, O., Leymann, F., Mitschang, B., Steinke, R.: Internet of things out of the box: using TOSCA for automating the deployment of IoT environments. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017), pp. 358–367. SciTePress, April 2017
10. Franco da Silva, A.C., Breitenbücher, U., Képes, K., Kopp, O., Leymann, F.: OpenTOSCA for IoT: automating the deployment of IoT applications based on the Mosquitto Message Broker. In: Proceedings of the 6th International Conference on the Internet of Things, pp. 181–182. ACM, November 2016



Optimizing Service Delivery with Minimal Runtimes

Katharina Gschwind¹, Constantin Adam^{2(✉)}, Sastry Duri²,
Shripad Nadgowda², and Maja Vukovic²

¹ Massachusetts Institute of Technology, Cambridge, USA
gschwind@mit.edu

² IBM T.J. Watson Research Center, New York, USA
{cmadam,sastry,nadgowda,maja}@us.ibm.com

Abstract. In this paper, we argue that deploying applications inside minimal runtime environments, which only contain the files necessary and sufficient for the application to run, can cut down the operating costs, specifically the costs for ensuring the application security and compliance. We identify a way to deliver minimal runtimes as Docker containers built from scratch. We describe a use case where minimal runtimes simplify the service maintenance operations, by reducing the number of updates for fixing security vulnerabilities.

1 Introduction

Keeping cloud applications secure is a complex process. Administrators need to check services and their underlying operating systems for vulnerabilities published on a daily basis by sources like Redhat security advisories [1], or Ubuntu security notices [2]. They also need to harden these services and their runtimes against threats using benchmarks such as those defined by the Center for Internet Security (CIS) [3]. In addition to the deployed application, these processes must be applied to the underlying OS, which in many cases is more complex than the application itself. The OS requires a large amount of configuration and updates, not necessarily related to the deployed application, to ensure that it cannot be used by an intruder to gain access to the Virtual Machine and/or Container and compromise it. Instead of implementing a process that secures the underlying operating system, we propose to eliminate it altogether.

In this paper, we argue that deploying applications inside minimal runtime environments, which only contain the code necessary and sufficient for the application to run, can cut down the operating costs, specifically the costs for ensuring the application security and compliance. We explore ways of automatically building and delivering such minimal runtimes, in the form of Docker containers built from scratch. We build containers from scratch for Redis, a popular open-source application, run the original and the minimal Redis image through a security vulnerability advisor, and show that minimal runtimes can reduce the number of re-deployments needed to keep up with various published security advisories.

2 Background

To secure an application, one must ensure that any vulnerabilities are remediated, and that the applications, as well as their underlying operating systems are configured properly. Vulnerability remediation and compliance enforcement are complementary actions that must be performed for several layers of software, depending on the type of the underlying runtime of a specific application. Figure 1 provides an illustration of the layers involved for virtual machines (cloud-enabled), regular containers (cloud-native), and minimal containers. Below, we describe in more detail the security and compliance procedures in place for each of these types of runtime.

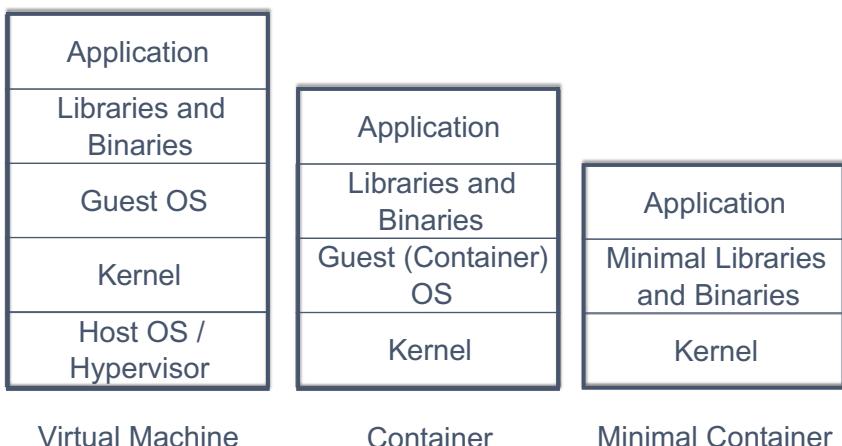


Fig. 1. Comparison of virtual machines, containers, and minimal containers.

2.1 Cloud-Enabled Applications

A VM running a cloud-enabled application today contains a full OS image (Linux or Windows) hosting a primary user space application (e.g., MySQL or Nginx), along with secondary services (e.g., SSH, syslog or NTP). These VMs contain multiple packages, services or drivers that are not needed by their primary application, but that increase their attack surface, and reduce their performance. Vulnerability remediations might require a change window, and not happen instantly. Performance-wise, patching can delay the system bootup (this happens frequently on Windows servers). Integration with other systems (user control, monitoring, inventory, patching, etc.) makes the task of enforcing compliance and remediating vulnerabilities on these VMs long and complex.

2.2 Cloud-Native Applications

For containers built from an operating system image, the inherent complexity of the underlying OS makes it hard for an administrator to know if a container is running compliant configurations, or non-vulnerable code. The OS and other services configuration is scattered across the file system, uses different formats; many packages installed in the container are inactive. The study in [4] shows that, in 2015, 64% of docker images in Docker hub had high profile vulnerabilities. The problem has been solved since then for the latest release of a majority of images, however, as new vulnerabilities appear, not only the latest release needs to be patched, but also all the already existing deployments. This is our main motivation to develop minimal runtimes. There is a common interest to reduce the trusted code base for application runtimes, with unikernels [5] advocating single address programming paradigms, or microcontainers [6] using statically compiled “go” applications or building containers from minimal Alpine images.

3 Implementing a Minimal Runtime

DockerSlim [7] is open-source software that creates minimal images including only the files necessary and sufficient to run specific applications. For each image, DockerSlim also creates Seccomp and AppArmor security profiles. It takes as input a “full” docker image built on top of an OS, and operates in three phases: initialization, monitoring, and image/security profile generation. During initialization, DockerSlim reverse engineers the dockerfile of the image provided, gathering information about volumes, exported ports, created users, etc. Next, it instantiates a container from the original image, modified to launch the docker-slim-sensor executable within. Finally, DockerSlim establishes communication channels to the instrumented container to send commands to the container, and receive monitored data. During monitoring, DockerSlim runs two sensors in the container: one tracks filesystem events and identifies the files and symbolic links needed to run the container, the other traces system calls and generates security profiles for the image. Finally, DockerSlim generates the minimal image and its security profile by processing the reports generated by the sensors.

4 Case Study

To demonstrate the advantages of a minimal runtime environment, we have downloaded a year-old ubuntu-based Docker image for Redis (version 3.2.2), and generated a minimal image using DockerSlim. We used the IBM Vulnerability Advisor (IBM VA [8,9]) to analyze both the original and minimal images for package-level security vulnerabilities, published in the Ubuntu security notices.

We had to make two changes for the IBM VA to function with images built from scratch. First, to identify the security advisory service against which IBM VA should check packages, we changed the DockerSlim code to include a file that contains information about the OS for which the files were built. Second, as

the IBM VA takes as input package information, we needed to map individual libraries to packages. With these changes, IBM VA was able to process slim images as though they were full images. Note that since IBM VA works at the package level, and a package consists of multiple files, IBM VA may report a vulnerability for a file that is not used in the minimal image. Further analysis of individual libraries in a slim image is needed to determine whether a package vulnerability is present in it or not.

We found that the original image had 10 vulnerabilities, in 6 vulnerable packages. The vulnerable packages were not necessary for the Redis application to function, therefore none of them was included in the minimal image generated from scratch. Deploying a minimal container, at the time of the release of the image, would have avoided dealing with 10 different security vulnerabilities, and potentially as many re-deployments of the Redis application.

5 Conclusion and Future Work

We have shown that minimal runtimes can cut down the number of updates that address security vulnerabilities. We will improve upon the DockerSlim method of generating images, and will conduct a large-scale evaluation of its security benefits on the set of Docker images with more than one million downloads. To ensure the completeness of the minimal runtime, we will add static analysis to the dynamic analysis provided by DockerSlim, and will ensure that test suites for the applications are automatically invoked during the building process.

References

1. Red hat customer portal security advisories. <https://access.redhat.com/security/security-updates/>. Accessed 04 Aug 2017
2. Ubuntu security notices. <https://www.ubuntu.com/usn/>. Accessed 04 Aug 2017
3. Center for internet security. <https://www.cisecurity.org/>. Accessed 04 Aug 2017
4. Van Tuin, C.: A security state of mind: compliance and vulnerability audits for containers. In: 2015 Usenix Container Management Summit, Washington, DC (2015)
5. Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., Crowcroft, J.: Unikernels: library operating systems for the cloud. In: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 461–472. ACM, New York (2013)
6. <https://www.iron.io/microcontainers-tiny-portable-containers/> . Accessed 15 Aug 2017
7. Dockerslim (docker-slim): optimize and secure your docker containers (free and open source). <https://dockerslim.im/>. Accessed 15 Aug 2017
8. Tak, B., Isci, C., Duri, S., Bila, N., Nadgowda, S., Doran, J.: Understanding security implications of using containers in the cloud. In: 2017 USENIX Annual Technical Conference (USENIX ATC 17), pp. 313–319. USENIX Association (2017)
9. Oliveira, F., Eilam, T., Nagpurkar, P., Isci, C., Kalantar, M., Segmuller, W., Snible, E.: Delivering software with agility and quality in a cloud environment. IBM J. Res. Dev. **60**(2–3), 10:1–10:11 (2016)



Semantic Data Mediator: Linking Services to Websites

Dennis Wolters^(✉), Stefan Heindorf, Jonas Kirchhoff, and Gregor Engels

Department of Computer Science, Paderborn University, Paderborn, Germany

{dennis.wolters,heindorf,engels}@uni-paderborn.de,
jonaskir@mail.uni-paderborn.de

Abstract. Many websites offer links to social media sites for convenient content sharing. Unfortunately, those sharing capabilities are quite restricted and it is seldom possible to share content with other services, like those provided by a user's favorite applications or smart devices. In this paper, we present *Semantic Data Mediator (SDM)* — a flexible middleware linking a vast number of services to millions of websites. Based on reusable repositories of service descriptions defined by the crowd, users can easily fill a personal registry with their favorite services, which can then be linked to websites by SDM. For this, SDM leverages semantic data, which is already available on millions of websites due to search engine optimization. Further support for our approach from website or service developers is not required. To enable the use of a broad range of services, data conversion services are automatically composed by SDM to transform data according to the needs of the different services. In addition to linking web services, various service adapters allow services of applications and smart devices to be linked as well. We have fully implemented our approach and present a real-world case study demonstrating its feasibility and usefulness.

Keywords: Services · Semantic data · Mediation · Data conversion
Interface adaptation

1 Introduction

Many websites enable users to share the presented content by integrating services of social media sites, e.g., the service to share content on Facebook or pin it on Pinterest. Mobile websites might even allow to share content via locally installed messengers like WhatsApp. Sharing content presented on a website with other services than those which are already linked to the respective site is up to end users, because website developers cannot accommodate all services relevant for users, especially since those might differ from user to user. Unfortunately, directly linking services to websites is a difficult task for end users, because it requires extensive technical knowledge. Thus, end users usually settle for manually copying content to other services, which is often very tedious.

For instance, when transferring a recipe found on a website to a cookbook app, all relevant information like title, ingredients, instructions, and associated images need to be copied to the respective fields of the cookbook app's input form. If the app does not run on the same device as the browser presenting the recipe, it is even more complicated.

In this paper, we explain how our approach Semantic Data Mediator [2] can be used to link additional services to websites. On the one hand, we demonstrate how regular end users without any programming knowledge can make use of SDM to link their favorite services to millions of websites. For this, SDM analyzes the semantic data embedded into websites for search engine optimization and typed over ontologies like schema.org¹. Thereupon, it determines which additional services can be offered for the found data. On the other hand, we explain the different building blocks of SDM and how power users can extend our approach. Moreover, we describe how we implemented SDM and provide a video demonstration how to use it. To the best of our knowledge, SDM is the first comprehensive approach enabling end users to bridge the gap between websites and services. Existing approaches are either restricted to a few selected services, e.g., share buttons for social media sites, or browser extensions for single services such as Skype.

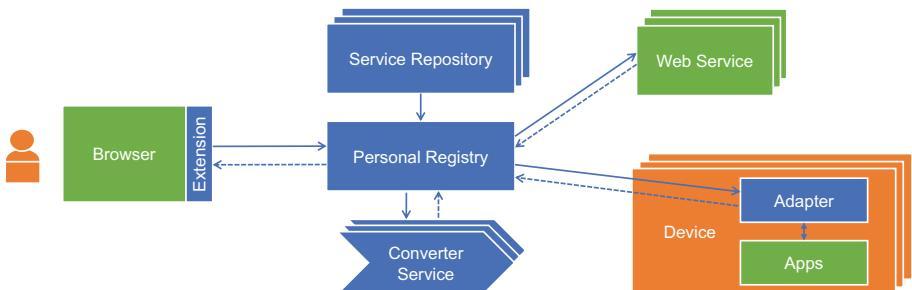


Fig. 1. Linking services to websites: architecture of semantic data mediator

2 System Overview

In this section, we give an overview of our approach *Semantic Data Mediator (SDM)*. The different buildings block of SDM visualized in Fig. 1 are explained in the following.

In the center of SDM is a *personal registry*, which manages all services relevant to a user. Services, which include both web services and services provided by locally installed applications, can be added to the registry by providing their OpenAPI descriptions². Those descriptions can either be referenced via an URL or can be retrieved from service repositories (see below). The personal registry

¹ <http://schema.org>.

² <https://github.com/OAI/OpenAPI-Specification>.

also contains services for data conversion, since the format of the semantic data embedded into websites usually differs from the data format expected by services. The input and output formats of such converter services are used to create a graph representing all possible conversion options. Based on this graph, the personal registry can efficiently compute output formats from given input formats. This information is used to determine which services can be offered to users.

A *service repository* contains OpenAPI specifications of services that can be added to a user's personal registry. Upon adding a service, credentials for using the service can be permanently stored or access tokens can be obtained, e.g., when OAuth is used. Additionally, user-specific parameters, e.g., an API key required to access a service, can be defined by using a custom property in the service description.

The *browser extension* extracts semantic data from a website and queries the personal registry for services that are able to process this data. These services are presented to the user either in an embedded menu attached to the visual representation of a data item on the website, or in a browser-based menu, which lists all data items embedded into the respective site. When a user chooses to invoke a service, the extracted data item is transmitted to the personal registry, which applies all necessary conversions using the previously determined converter services and afterwards transmits the (converted) data to the chosen service. A service can be a web service or it can be a service provided by an application installed on one of the user's devices. This does not necessarily has to be the device on which the browser runs, since SDM supports the cross-device integration of services via adapters.

Adapters are used to create external service interfaces for applications that only provide interfaces limited to other applications within the device on which they are installed. For instance, we use our Cross-Device Application Integration approach for Android apps (XDAI-A) [3,4] to enable the usage of services offered by Android apps, e.g., sharing data with a messenger app, adding data like an event, a recipe, or a note to apps like Android Calendar, MyCookbook³, or Google Keep⁴, respectively. XDAI-A provides a domain specific language to create adapters to support further Android apps. Additionally, we provide a customizable command-line interface (CLI) adapter that can be used to enrich command-line tools with a RESTful HTTP interface. To configure the adapter, an OpenAPI specification needs to be created, which describes the different endpoints provided by the service. For each endpoint, a custom property is used to configure which command-line tool is executed upon incoming requests. Data from an HTTP request can easily be mapped to arguments of the invoked tool and the tools output can again be mapped to the HTTP response. We also provide a Cross-Device Application Integration approach for Windows applications (XDAI-W), which extracts information about installed applications from

³ <http://mycookbook-android.com/>.

⁴ <https://google.de/keep/>.

the Windows registry and offers an HTTP interface to use these applications in combination with SDM.

Converter services are specially tagged services that are used by the personal registry to convert the semantic data extracted by the browser extension. SDM can automatically compose converter services, and thereby, enables the use of services that only accept a data format different to the one of the extracted data. The data conversion takes both the semantic type, e.g., event or recipe, as well as the data format, e.g., JSON-LD or RDF/XML into account. Details about the data conversion can be found in [2]. To accelerate the creation of converter services, we provide a template which only requires a conversion function and some meta information, like a description and input/output types. Additionally, our CLI adapter allows to simply reuse existing command-line conversion tools as converter services. For instance, we use this adapter to provide a converter service based on the universal document converter Pandoc⁵.

In combination, the previously described building blocks of SDM allow to link services to the millions of websites that provide semantic data [1]. Further support for SDM by website developers or service providers is not required. SDM is simple enough to be used by average end users, while power users can also extend SDM by adding new (converter) services or adapters.

3 Implementation Details

The browser extension is available for Google Chrome. Similar extensions can be developed for all major browsers since no features unique to Chrome have been used to implement the extension. The converter service template, the CLI adapter, the Windows adapter, the server-side part of the Android adapter, and the personal service registry (see Sect. 2) are developed using JavaScript and the Node.js runtime. The client-side part of the Android adapter consists of two interconnected apps, one responsible for providing the external interface and the other one for adapting service request. Further details on the Android adapter are given in [3,4]. The personal registry uses a document-oriented database (CouchDB) for storing OpenAPI specifications of all registered services along with some additional parameters, like authentication tokens or credentials to use the services. Furthermore, it uses a graph database (Neo4j) to efficiently find conversion options and relevant services for a given website. SDM natively supports OpenAPI service descriptions. Other service description languages, like RAML⁶ or API blueprint⁷, are supported by converting them upfront to OpenAPI.⁸ To enable easy deployment, the registry as well as all converters based on our converter service template or CLI adapter can be deployed as Docker containers.

⁵ <https://pandoc.org/>.

⁶ <http://raml.org>.

⁷ <http://apiblueprint.org/>.

⁸ <http://github.com/LucyBot-Inc/api-spec-converter>.

4 Demonstration

The demonstration consists of two parts: (i) Usage of SDM and (ii) technical details.⁹ In the first part, we show how SDM links services to websites and we demonstrate the invocation of web services as well as services provided by applications installed on different devices. For instance, we demonstrate how a recipe found on a website on a desktop computer can be exported to Word or can be added to a cookbook app on a smartphone. Moreover, we explain the addition of services to the personal registry by utilizing services repositories. In part two, we explain the technical details of SDM. We show how the personal registry maintains the information about possible data conversions and how this information is exploited to find services that require data in a specific format. Further, we present the different types of adapters and explain how they can be used to create new services that are accessible for SDM.

References

1. Bizer, C., Meusel, R., Primpeli, A.: Web Data Commons - RDFa, Microdata, and Microformat Data Sets. <http://webdatacommons.org/structureddata/>
2. Wolters, D., Heindorf, S., Kirchhoff, J., Engels, G.: Linking services to websites by leveraging semantic data. In: ICWS 2017, pp. 668–675. IEEE (2017)
3. Wolters, D., Kirchhoff, J., Gerth, C., Engels, G.: Cross-device integration of android apps. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 171–185. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46295-0_11
4. Wolters, D., Kirchhoff, J., Gerth, C., Engels, G.: XDAI-A: framework for enabling cross-device integration of android apps. In: Drira, K., et al. (eds.) ICSOC 2016. LNCS, vol. 10380, pp. 203–206. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68136-8_25

⁹ A demo video is available at: <http://sdm.dwolt.de/demo/>.



ARGON: A Tool for Modeling Cloud Resources

Julio Sandobalin^{1,2()}, Emilio Insfran², and Silvia Abrahao²

¹ Escuela Politécnica Nacional, Ladrón de Guevara, E11-253,
P.O. Box 17-01-2759, Quito, Ecuador

julio.sandobalin@epn.edu.ec

² Universitat Politècnica de València, Camino de Vera, s/n,
46022 Valencia, Spain

{jsandobalin, einsfran, sabrahao}@dsic.upv.es

Abstract. Configuration Management Tools (CMTs) have achieved automate the orchestration of infrastructure provisioning in the Cloud. However, CTMs are managed through script languages in a manually intensive manner. As a result, CMTs management is a time-consuming and error-prone activity. In a previous work, to face this issue, we have presented ARGON tool which models the cloud infrastructure and generate scripts for managing the CMTs. In this paper, we propose an extension of ARGON tool with functionalities both to modeling computing, storage, networking and elasticity in the Cloud and to model the middleware needed for the proper operation of cloud-based software applications.

Keywords: Cloud computing · DevOps · Infrastructure as code
Cloud infrastructure provisioning · Model-Driven development

1 Introduction

DevOps (Development & Operations) is a paradigm which brings principles, practices and tools that optimize the software delivery time. The cornerstone of DevOps is the Infrastructure as Code [1] that is infrastructure automation based on practices from software development. In the DevOps community there exist several Configuration Management Tools (CMTs) that orchestrate the infrastructure provisioning automation in the Cloud, such as Ansible¹, Chef² or Puppet³. Each of CMTs works with specific cloud providers. However, CTMs define the infrastructure provisioning through script languages in a manually intensive manner. As a result, use scripts for defining the infrastructure provisioning is a time-consuming and error-prone activity. In a previous work, to face this issue, we have presented ARGON [2], which is a tool developed for abstracting the complexity of infrastructure modeling for different cloud providers through a Domain Specific Language. Furthermore, ARGON abstracts the features of scripting languages of the CMTs to define transformation rules, which are used to

¹ <https://www.ansible.com>.

² <https://www.chef.io>.

³ <https://puppet.com>.

generate scripts for managing the CMTs. As a first approach, ARGON supports the resource modeling of Infrastructure as a Service (IaaS) for Amazon Web Services.

In this paper, we propose an extension of ARGON tool with functionalities both to modeling computing, storage, networking and elasticity in the Cloud and to model the middleware needed for the proper operation of cloud-based software applications.

On the other hand, ARGON tool is the keystone of our approach of an end-to-end automated toolchain for cloud infrastructure provisioning [3].

2 An Infrastructure Modeling Tool for Cloud Provisioning

ARGON is a modeling tool for specifying the final state of the infrastructure provisioning in the Cloud and generate scripts for managing Configuration Management Tools (CMTs). ARGON has a Domain Specific Language (DSL) that includes an abstract syntax and a concrete syntax. The former is an Infrastructure Metamodel [2], which abstracts the capacities of the Cloud Computing, such as computing, storage, networking and elasticity. The latter defines graphical notation to render the metamodel elements in the modeling editors of Eclipse Modeling Framework [4]. Moreover, ARGON includes a transformation engine based on model-to-text (M2T) transformations to generate scripts for managing the CMTs.

ARGON use the DSL to support the infrastructure modeling of different cloud providers. The Infrastructure Metamodel [2] propose a generic solution for modeling the cloud infrastructure. However, to model the infrastructure of a particular cloud platform, it is necessary to make model-to-model (M2M) transformations in order to get a particular infrastructure model, for instance, Microsoft Azure infrastructure model or Amazon Web Services infrastructure model. Once the infrastructure model is finished, the transformation engine uses it and execute model-to-text transformations for generating scripts for a specific CMT, for instance, creating a playbook for Ansible or recipe for Chef.

Additionally, ARGON can work with Maven to develop infrastructure projects. Finally, because the ARGON's components were developed following a plugin-based architecture, they can be used in Eclipse Modeling Framework or an integration server (e.g. Jenkins).

Table 1. Functionalities to model elasticity capacity

Figure	Element	Description
	Launch Configuration	Specify a template with hardware features of a Virtual Machine
	Auto Scaling Group	Specify the number of Virtual Machines to create or terminate
	Scaling Policy	Specify the conditions to create or terminate a Virtual Machine
	Alarm	Monitor a metric in a time frame to trigger a Scaling Policy

2.1 Functionalities

The functionalities of the ARGON's elements are described in Tables 1, 2, 3 and 4. The functionalities to model networking capacity is represented by associations among metaclasses specified in the Infrastructure Metamodel [2].

Table 2. Functionalities to model storage capacity

Figure	Element	Description
	Database	Create, terminate, start and stop Databases instances
	File Server	Create and terminate File Servers

Table 3. Functionalities to model computing capacity

Figure	Element	Description
	Virtual Machine	Create, terminate, start and stop virtual server instances
	Security Group	Perform as a firewall to protect Virtual Machines
	Inbound	Specify inbound rules to enforce in Security Groups
	Outbound	Specify outbound rules to enforce in Security Groups
	Static IP	Allocate or release a static IP address to a Virtual Machine
	Load Balancer	Register or unregister a virtual machine to a Load Balancer
	Listener	Check the connection requests to a Load Balancer

Table 4. Functionalities to model middleware

Figure	Element	Description
	Role	Specify software packages to be installed as middleware
	Deployment	Specify path, permission, and user to install software applications
	Template	Specify configuration files to be replaced in a virtual server

3 Modeling Infrastructure and Middleware

ARGON allows modeling a scalable architecture to provide the elasticity capacity to cloud-based software applications. Figure 1a shows an infrastructure model that has a **Launch Configuration** with hardware features of a virtual machine. **Auto Scaling Group** specify the minimum and maximum number of virtual machines to be created or

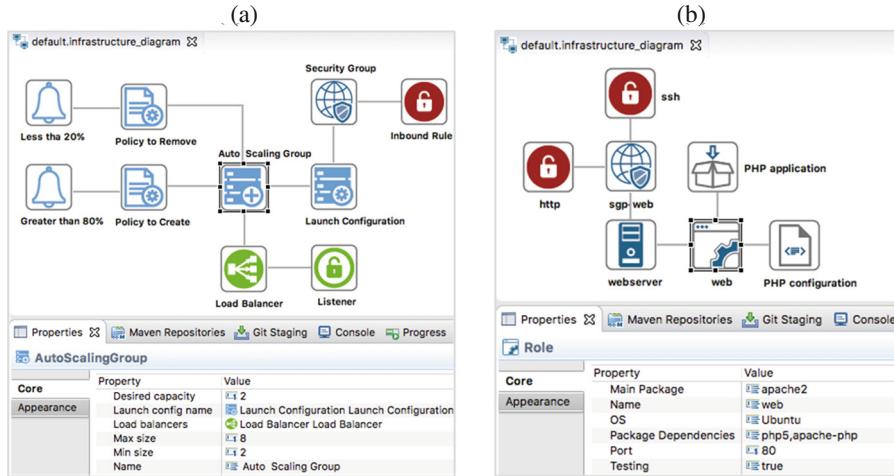


Fig. 1. (a) Scalable architecture for cloud infrastructure provisioning. (b) Modeling middleware and software applications.

terminated according to hardware features from **Launch Configuration**. Creation of a virtual machine is done based on **Policy to Create** which is executed by reaching **Greater than 80%** of CPU usage of a virtual machine. Termination of a virtual machine is done based on **Policy to Remove** which is executed by reaching **Less than 20%** of CPU usage of a virtual machine. **Security Group** performs as a firewall to all virtual machines created in the scalable architecture. **Security Group** enables the virtual machines to respond requests through ports as **Inbound Rules**. **Load Balancer** allows distributing incoming application traffic between multiple virtual machines and with an input rule or **Listener** checks the connection requests.

On the other hand, ARGON allows modeling the middleware and cloud-based software applications. Figure 1b shows a virtual machine (**webservice**) which has a security group (**sgp-web**) to enable the virtual machine to respond requests through port 22 (**ssh**) and port 80 (**http**).

To model the middleware, we define a role (**web**) in which we specify software packages to be installed in the virtual machine (**webservice**), kind of operative system, port to respond requests to the virtual server, and the option to testing the infrastructure deployed. Moreover, ARGON allows modeling the deployment (**PHP application**) of software applications and specify a template (**PHP Configuration**) or configuration file that will be replaced in the virtual server.

Acknowledgments. This research is supported by the Value@Cloud project (TIN2013-46300-R).

References

1. Morris, K.: *Infrastructure as Code: Managing Servers in the Cloud*, 1st edn. O'Reilly Media, Inc. (2016)
2. Sandobalin, J., Insfran, E., Abrahao, S.: An infrastructure modelling tool for cloud provisioning. In: Proceedings of 14th IEEE International Conference on Services Computing, SCC (2017). (In press)
3. Sandobalin, J., Insfran, E., Abrahao, S.: End-to-End automation in cloud infrastructure provisioning. In: Proceedings of 26th International Conference on Information Systems Development, ISD (2017). (In press)
4. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: eclipse modeling framework (2008)



Ubiquity: An Extensible Framework for Persistence in Container Environments

Mohamed Mohamed^(✉), Robert Engel, Amit Warke, and Heiko Ludwig

Almaden Research Center, IBM Research, San Jose, CA, USA

{mmohamed,engelrob,aswarke,h ludwig}@us.ibm.com

Abstract. Within the last few years, containers are being used for a broad set of applications, many of which have extensive requirements in terms of persisting their state. These stateful applications are still not well supported in container-based environments due to the challenges of adding persistence support. There are many efforts being done recently to tackle these challenges but most of them are focused on one environment or one storage provider. In this paper, we present the Ubiquity framework, which provides an extensible way of provisioning persistent storage for stateful containers. Ubiquity can be used to provide different types of persistent volumes from heterogeneous providers to be consumed by heterogeneous container frameworks in a seamless manner.

Keywords: Persistence · CloudFoundry · Kubernetes · Docker

1 Introduction

In the last few years micro-services became the new trend for designing software applications. In this paradigm, software applications are developed as a set of independent components that focus on small functionalities, can be deployed separately, and use some lightweight communication mechanism such as REST, gRPC, etc. These components could be easily deployed in containers that present a lightweight operating system level virtualization mechanism where the application running inside the container shares the kernel with the host operating system but has its own root file system [5]. Many platforms and orchestration systems are based on container concepts to offer an agile way of building micro-service based software such as Cloudfoundry, Docker, Kubernetes, Mesos, etc. While using these container orchestrators (COs) is beneficial for large scale deployments, there is still a lively discussion as to which type of applications they might be best suited for. Most of these COs favor stateless micro-services due to the challenges of managing state in concurrency situations.

Onboarding stateful applications into these COs in a scalable way is not well supported particularly in scenarios where state is accessed from workloads in different deployment platforms. In these scenarios, even though we can have

the same persistence backend used to maintain the state, the frameworks that are being used have heterogeneous ways of consuming persistence storage. For example, CloudFoundry uses persistent volumes through its persistence drivers and service brokers, whereas Kubernetes has a dynamic provisioner responsible for the creation and deletion of volumes and a volume plugin responsible for the other consumption functions (e.g., attach/detach, mount/unmount). The other challenge is that the heterogeneous management functionalities for persistent storage may change from one provider to another and from one persistent volume type to another (filesystem or block device based).

In this demonstration paper, we present the Ubiquity framework that enables seamless access to storage for heterogeneous COs. Ubiquity integrates with the widely used orchestrators (CloudFoundry, Kubernetes, Docker, Mesos, OpenShift). It also provides an easy way of adding new storage backends without the need to understand the specificities of the COs that are supported. We will demonstrate how Ubiquity can be used in different scenarios to provide different types of persistent storage in heterogeneous environments. In Sect. 2, we will present Ubiquity and its different components. Afterwards, we will give different use cases and demonstrate how ubiquity can help managing the persistent volumes for the used COs in Sect. 3. Then, we conclude the paper in Sect. 4.

2 Overview of Ubiquity Framework

Ubiquity framework allows COs operators to offer persistent storage from various providers without the need to understand the intrinsics of storage backends. At the same time, it allows storage providers to offer their persistent storage to containers without understanding the intrinsic requirements of the different COs. As shown in Fig. 1, Ubiquity is made up of different loosely coupled components that are easy to extend or replace. We briefly introduce these components in the following subsections.

Ubiquity Volume Service: This service is the main component of Ubiquity playing the role of the mediator between the COs and the storage backends. It is offering southbound interfaces to be consumed by COs to allow them to create persistent volumes and manage them. The management operations are continuously evolving based on the functionalities supported by the COs. These operations include attaching/detaching volumes to nodes, setting quota/size of volumes, maintaining the coherence of the attachment of volumes.

Container Orchestrators Plugins: Ubiquity framework has support for different COs using their specific storage plugins. The plugins generally live in all the nodes managed by the CO and they allow to execute host side operations to make the persistent storage ready to be consumed by the containers. So far, we have support for persistence for CloudFoundry (through an implementation of the Open Service Broker API [2]), Docker (through an implementation of the docker plugin API [4]), and Kubernetes environments (through a dynamic provisioner [1] and a volume plugin implementing the Flex Volume API [7]).

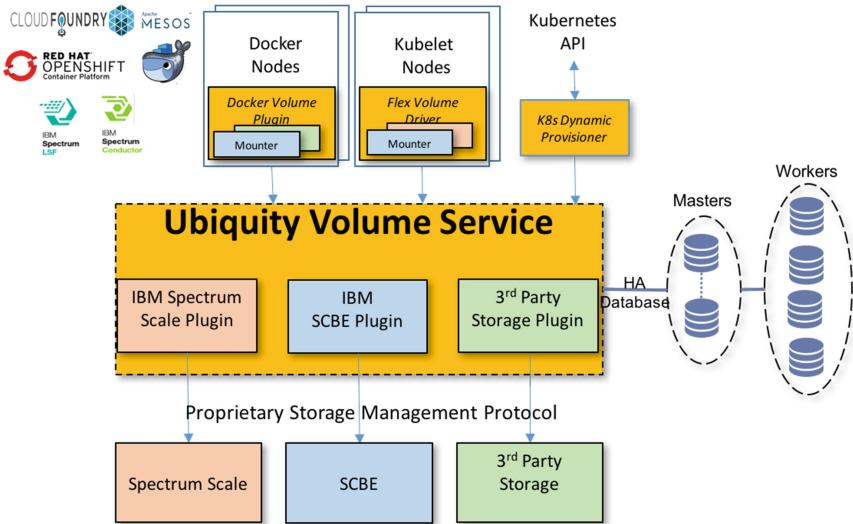


Fig. 1. Ubiquity architecture overview

Storage Backends: These are the mechanisms needed from the storage provider perspective to make their storage ready to COs. These components implement the needed mapping between Ubiquity API and the specific provider API to allow the creation and management of persistence storage.

Database Service: This service is used for our locking mechanism to manage concurrent access to volumes. It is also used to enable high availability based on leader election algorithms.

3 Demonstration

In this section, we will describe the two demos showing how to use Ubiquity in different scenarios.

Shared volumes: In this demo, we will show how Ubiquity allows providing persistent volumes across different container environments. We have a multi-tier application where the visual part of the application runs in CloudFoundry and the processing part runs in docker. The application on CloudFoundry shows a catalog of pictures saved in a persistent volume bound to the application. We start a docker deployment using the same persistent volume to run some face recognition algorithms using OpenCV. We show how the content is changed in the CloudFoundry side as well. In the demo, we will show the detailed steps towards enabling both environments to share the same persistent volume. The persistent storage is created out of distributed filesystem.

Dedicated volumes: In this demo, we show how we create a dynamic Cassandra cluster on kubernetes. Since Cassandra uses its own filesystem, we need to use persistent volumes based on block devices. One challenge here is to make each instance of the cluster get its own block device to avoid data corruption. Whenever a new instance is added (scale up), a new persistent volume is created and mounted into the new pod. Ubiquity allows to do that by creating a storage class that refers to our dynamic provisioner. We refer to the storage class in the persistent volume claim template related to the descriptor of the Cassandra container. Scaling the cluster up or down becomes a straight forward action backed up with Ubiquity. We can easily use Cassandra from any instance and check that the data is replicated and well maintained. If ever we loose a pod, kubernetes will recreate it and Ubiquity will ensure that the pod is bound to the right persistent volume to maintain the consistency of the cluster. In this demo, we can simulate a failure by killing one or more cluster nodes. Kubernetes will recreate the nodes and Ubiquity will make sure that the data will be recovered.

4 Conclusions

Using containers to deploy stateful applications is a challenging task. Given that each container orchestrator (CO) has a different set of APIs and storage features, the burden becomes on the storage vendor to integrate into each CO. So far there are little efforts being done to integrate this diverse set of COs and storage systems. Efforts such Rexray [3], Trident [8] and Torus [6] are either specific to one framework or one storage provider. So far, none of them offers support for Cloudfoundry, nor heterogeneous support for different container environments and different storage providers at the same time. In this demonstration paper, we proposed the Ubiquity framework that addresses the complexity of bringing together the different COs and Storage providers in the context of providing persistent storage across COs.

References

1. Dynamic provisioning and storage classes in kubernetes. <http://blog.kubernetes.io/2016/10/dynamic-provisioning-and-storage-in-kubernetes.html>
2. Open service broker API. <https://github.com/openservicebrokerapi/>
3. Rex-ray openly serious about storage. <https://rexray.readthedocs.io/en/stable>
4. Volume plugins. https://docs.docker.com/engine/extend/plugins_volume
5. Azab, A.: Enabling docker containers for high-performance and many-task computing. In: IC2E (2017)
6. Michener, B.: Presenting torus: a modern distributed storage system by coreos. <https://coreos.com/blog/torus-distributed-storage-by-coreos.html>
7. Nelluri, C.: Flexvolume explored. <https://www.diamanti.com/blog/flexvolume-explored/>
8. Sullivan, A.: Introducing trident: a dynamic persistent volume provisioner for kubernetes. <http://netapp.io/2016/12/23/introducing-trident-dynamic-persistent-volume-provisioner-kubernetes>



Distributed Video Analytics Across Edge and Cloud Using ECHO

Aakash Khochare, Pushkara Ravindra^(✉), Siva P. Reddy,
and Yogesh Simmhan

Indian Institute of Science, Bangalore 560012, India
`{aakhochare,kommareddy}@grads.cds.iisc.ac.in, pushkar1593@gmail.com,`
`simmhan@iisc.ac.in`

Abstract. Analytics over urban video streams is well suited for distributed computing across Edge, Fog and Cloud. Such streams are network intensive, making it prohibitive to fully transfer them to the Cloud. Deep Neural Networks have achieved remarkable accuracy in image classification, but are computationally costly on just Edge devices. We propose ECHO as a big data platform to compose IoT dataflows and seamlessly distribute them across Edge and Cloud resources. In this demonstration, we illustrate the capabilities of ECHO for deploying several video analytics applications to support smart city use-cases.

1 Introduction

Internet of Things (IoT) is proliferating sensing and actuation devices in the physical space around us. Smart Cities, a manifestation of IoT, use analytics over streaming data sensed from city infrastructure to make management decisions on public utilities, traffic control, public safety, etc. While such processing has traditionally been limited to either local computation at the data source or centralized computation in the Cloud, analytics over video streams from thousands of cameras in a city challenges these two exclusive approaches.

The rise of deep neural network models is radically advancing computer vision algorithms to match humans in their ability to classify images. Such models can transform video streams into a urban meta-sensor to detect traffic movement, people density, pollution levels, safety violations, etc. But model inferencing is computationally costly, often requiring GPU acceleration, with model training even costlier. The typical approach of moving all the data to the Cloud for scalable analytics is bandwidth-intensive for video streams, and introduces network latencies during decision making. Further, such models are just one part of more complex applications that perform pre-processing and decision-making too.

The availability of distributed Edge and Fog devices as part of smart city deployments with substantial cumulative computing capacity can be leveraged in conjunction with Cloud resources for such urban video analytics applications. This requires an application platform to compose these dataflows, deploy

them on distributed resources, and seamlessly manage their online orchestration. ECHO is one such platform that we have developed to address these needs [1].

In this demo, we showcase the ability of the ECHO platform to deploy and manage urban video analytics applications across Edge, Fog and Cloud resources.

2 Background and Related Work

ECHO¹ is a platform for *Orchestration of Hybrid dataflows across Cloud and Edge* [1]. It allows the user to compose applications as a dataflow of tasks, with support for *hybrid data models* such as streams, micro-batches and files flowing through. An *application manager* deploys these tasks on distributed Edge, Fog and Cloud resources, using a *platform service* that runs on each device. A *scheduler* maps tasks to resources based on their availability maintained in a *registry*. Once deployed, the tasks are orchestrated by an *Apache NiFi* engine on each resource, which we extend for distributed execution. We also support delegation of parts of the dataflow to *external native engines* like *Apache Edgent* for Complex Event Processing (CEP), *Apache Storm* for distributed stream processing, and *Google TensorFlow* for deep learning. ECHO incorporates *dynamic adaptation* to remap tasks onto different resources, on-demand, to meet an application's current needs.

Other IoT middleware offer a limited subset of these capabilities. *Apache Edgent* supports CEP querying over event streams on Edge Devices [2]. It is however not designed for distributed execution across multiple devices. *Node.RED* [3] and native *NiFi* provide interactive dataflow composition across edge, fog and cloud machines. While ECHO has a similar dataflow model, it automates the deployment of the dataflow tasks across distributed resources with awareness of network asymmetry, supports files, stream and micro-batches as data exchange formats between tasks, and also allows a pluggable scheduler logic to determine task placement. Amazon AWS's *GreenGrass* and Microsoft's *Azure IoT SDK* have started offering edge and fog device management interfaces to complement their Cloud services [4, 5]. AWS also supports composable serverless lambda functions to be deployed on edge and Cloud devices. ECHO goes beyond these capabilities and allows complex user logic to be embedded in the tasks, allows interfacing with external Big Data platforms like Storm, Edgent and TensorFlow, and offers a non-proprietary platform that can be extended by users.

3 Extensions to the ECHO Platform

In this demo, we extend ECHO with two novel features that we discuss next: *efficient scheduling* and managing *network asymmetry*.

Resource and Energy-Aware Scheduling. We have earlier proposed the scheduling of a given dataflow onto Edge and Cloud resources as an optimization

¹ <https://github.com/dream-lab/echo>.

problem and solved it using a *Genetic Algorithm (GA) meta-heuristic* for an individual directed acyclic graph (DAG) [6]. Here, we extend this meta-heuristic approach for scheduling to support dataflows that arrive and depart continuously within the Edge, Fog and Cloud resources, and integrate the scheduler algorithm with **ECHO**. The optimization problem takes the tasks, their compute latencies, throughput, and energy footprint on different devices, and the network latency and bandwidth between devices as input. It enforces constraints to prevent the compute capacity for a single device from being saturated, and the energy usage on an edge device from draining its battery before it is recharged. The meta-heuristic scheduler has the goal of reducing the dataflow's end-to-end latency subject to the aforementioned constraints.

We extend and use this for **ECHO**'s adaptive scheduler. The *app manager* passes the user's DAG to the *scheduler*, along with the state of available resources from the registry. The *scheduler* produces a mapping of tasks to devices while meeting the optimization goal. The algorithm can later be rerun for adaptive re-balancing in case the dataflow's latency does not match the requirements due to factors such as increase in the input rate. We propose to demonstrate the meta-heuristic scheduler and rebalancing.

Managing Network Asymmetry. **ECHO**'s app manager invokes the REST platform service on each device to deploy and connect the dataflow tasks. However, this requires that the manager service on the Cloud be able to access the platform service on every resource over the Internet. Edge and Fog devices are often behind firewalls, making them inaccessible from the public Internet. Here, we mitigate this by extending the app manager to support asynchronous message passing to the platform service using an MQTT publish-subscribe broker. The platform service in each resource subscribes to a unique topic in the broker to which the manager publishes control messages for initiating a dataflow deployment. Each deployment session spawns a unique topic which is used to pass request and response JSON messages. Through this pattern, only the broker needs to be in a network location that is visible to all devices. Such a network asymmetry can also affect tasks on different devices that need to pass data items. Besides the existing support for both a push and a pull mechanism between two NiFi engines, we further support a similar broker-based model using Apache Kafka for scalable transfer of large and fast data streams within the application. We will demonstrate support for such forms of network asymmetry.

4 Video Analytics Applications

We design two representative video analytics dataflows motivated by smart city applications, and demonstrate their execution using **ECHO** on Edge, Fog and Cloud resources. Our IoT testbed (Fig. 1(Bottom)) where these applications are deployed consists of 12 Raspberry Pi 2B and 3B edge devices, an NVIDIA TX1 and a SoftIron ARM64 Fog servers, four Azure DS1 VMs in Microsoft's South India data center, and one Azure NC6 GPU VM in the US East data center [1].

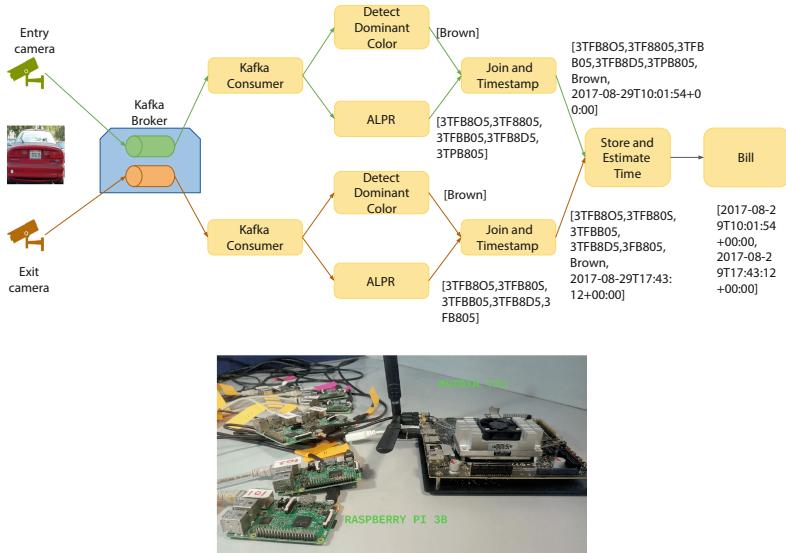


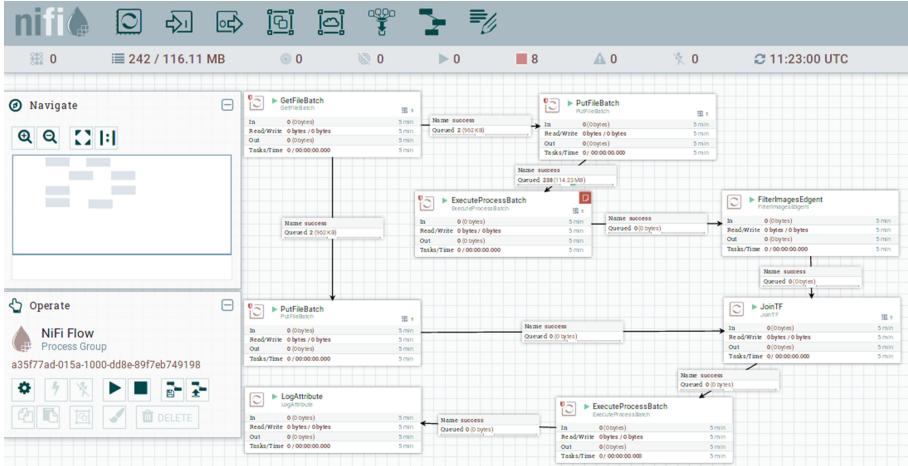
Fig. 1. ALPR dataflow for parking billing (Top), and IoT Testbed Devices (Bottom)

Automatic Billing of Parking. Automatic License Plate Recognition (ALPR), a popular computer vision analytic, is used in applications like traffic enforcement, congesting pricing and automated toll collection. It is solved in two parts – the license plate region is first detected in an image, and then the characters are extracted from the region using Optical Character Recognition (OCR) [7]. Here, we design a dataflow that uses ALPR for automated time-based billing of vehicles across hundreds of parking lots in a city, when cameras are present at their entry and exit gates. We correlate the time at which a license plate enters and when it exits using ALPR, and bill them on exit. The challenge comes from the ALPR algorithm giving false positives. To address this, we also capture and store the detected color of the vehicle², besides the top n estimates of the license plate returned by an OpenALPR task³ and the timestamp when a vehicle enters. When a vehicle exits, these image analytics algorithms are rerun to detect its color along with the estimated plate numbers by ALPR, which are compared using a distance function with the details of vehicles that entered earlier. A match is found if the best distance score is above a threshold, and is used to determine the duration of parking and the bill. This dataflow that will be demonstrated is shown in Fig. 1(Top).

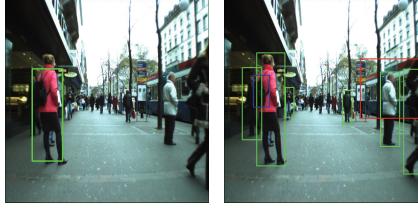
Urban Scene Classification. Classification algorithms based on deep learning models associate bounding-boxes and tags to different entities in a given image. The outputs from such models can be used to detect situations of interest in urban environments, such as safety incidents, traffic violation, etc. YOLO is

² <https://github.com/fengsp/color-thief-py>.

³ OpenALPR library, <https://github.com/openalpr/openalpr>.



(a) YOLO dataflow deployed within NiFi



(b) YOLO Tiny (c) YOLO Full

Fig. 2. YOLO dataflow and classified outputs from models

one such popular deep convolutional neural network for object detection that is trained and available on TensorFlow [8]. We demonstrate a novel use of YOLO using a two-level classification of urban scenes in conjunction with an *Apache Edgent CEP engine*, as described in [1]. Running YOLO on a full resolution image frame (608×608) is computationally costly, and TensorFlow achieves a frame-rate of only 1/s even with an NVIDIA K80 GPU. Instead, we use an additional *tiny* YOLO model that operates on a scaled-down image on the Edge or Fog device, and if any interesting tags are detected, forwards a full-resolution video segment to a GPU VM on the Cloud where the full model runs for accurate classification. This also illustrates the use of hybrid engines, TensorFlow and Edgent, for execution within ECHO . A screenshot of the dataflow in NiFi, along with a sample frame classification from the models is shown in Fig. 2.

5 Demonstration

The ECHO platform's features and capabilities will be demonstrated by deploying the two applications elaborated in Sect. 4.

Automatic Billing of Parking. We treat the input video as a stream of distinct images. We source these images from Caltech automobile dataset [9] and periodically publish them into either the entry camera topic or the exit camera topic hosted in a *Kafka Broker*. As Fig. 1(Top) shows, the first processors in the dataflow are *Kafka Consumers* that acquire these image streams. This application will be deployed using the *heuristic scheduler* onto our IoT testbed. This results in a mapping of tasks to the devices, such that the optimization goals are met. We will then use the deployed application along with a custom scheduler that will deterministically produce a different mapping to demonstrate the *rebalance* capability of ECHO.

Urban Scene Classification. In this application too, we consider the input video stream as a sequence of images. The images are located in the file system of the edge device which serves as the input of the video. We use images from the ETH people dataset as exemplar [10]. The *heuristic scheduler* is used for the placement of this dataflow as well. We should see a placement where the native TensorFlow processor is mapped onto the GPU accelerated fog device.

In both these demonstrations, we will demonstrate the platform seamlessly handling the network asymmetry that exists between the devices, where some devices may not be able to access others. Data is either pushed or pulled from the source to the destination, or vice versa, depending on this network visibility. If neither can reach each other directly over the network, a Kafka Message Broker is used to pass the data through indirection.

References

1. Ravindra, P., Khochare, A., Reddy, S.P., Sharma, S., Varshney, P., Simmhan, Y.: ECHO: an adaptive orchestration platform for hybrid dataflows across Cloud and Edge. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 395–410. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_28
2. Apache Edgent, v1.1.0. <http://edgent.apache.org/>. Accessed 21 June 2017
3. Node-RED. <https://nodered.org/>. Accessed 1 Oct 2017
4. Amazon AWS Greengrass. <https://aws.amazon.com/greengrass/>. Accessed 21 June 2017
5. Microsoft Azure IoT Edge. <https://azure.microsoft.com/en-in/campaigns/iot-edge/>. Accessed 21 June 2017
6. Ghosh, R., Simmhan, Y.: Distributed scheduling of event analytics across Edge and Cloud. ACM Trans. Cyber-Phys. Syst. (2017, To appear)
7. Ozbay, S., Ercelebi, E.: Automatic vehicle identification by plate recognition. World Acad. Sci. Eng. Technol. **9**(41), 222–225 (2005)
8. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger, arXiv preprint [arXiv:1612.08242](https://arxiv.org/abs/1612.08242) (2016)
9. CalTech Cars Dataset. <http://www.vision.caltech.edu/archive.html>. Accessed 28 Aug 2017
10. Ess, A., Leibe, B., Schindler, K., van Gool, L.: A mobile vision system for robust multi-person tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)

Author Index

- Abdellatif, Takoua 43
Abraha, Silvia 393
Adam, Constantin 384
Aiello, Marco 174
Anderson, Maya 198
Ayed, Rahma Ben 80
- Badr, Youakim 93
Belala, Faiza 5, 111
Benzadri, Zakaria 111
Beranek, Marek 258
Bermbach, David 198, 240
Berrocal, Javier 124
Boubeta-Puig, Juan 228, 369
Bozga, Marius 43
Breitenbücher, Uwe 379
- Calegari, Daniel 308
Canal, Carlos 124
Carrasco, Jose 55
Clarke, Siobhán 149
Corral-Plaza, David 369
- de Oliveira, Patrícia Araújo 363
Delgado, Andrea 308
Dong, Yuji 137
Dubosson, Magali 18
Durán, Francisco 55, 363
Duri, Sastry 384
- Engel, Robert 398
Engels, Gregor 388
- Fay, Alexander 270
Ferme, Vincenzo 215
Fernandez, Pablo 352
Feuerlicht, George 258
Flores-Martin, Daniel 333
Fragnière, Emmanuel 18
Franczyk, Bogdan 252
- Gamez-Diaz, Antonio 352
Garcia-Alonso, Jose 124
- García-de-Prado, Alfonso 228
Gatouillat, Arthur 93
Geiger, Matthias 215
González, Laura 308
Graja, Imen 67
Gschwind, Katharina 384
Guerfel, Rawand 80
Guermouche, Nawal 67
- Hadj-Alouane, Nejib Ben 161
Hameurlain, Nabil 5
Harrer, Simon 215
Haubbeck, Christopher 270
Heindorf, Stefan 388
- Insfran, Emilio 393
- Junod, Nathalie 18
- Kacem, Ahmed Hadj 67
Kallel, Slim 67
Kat, Ronen 198
Képes, Kálmán 379
Khebbab, Khaled 5
Khochare, Aakash 402
Kirchhoff, Jonas 388
Kitouni, Roumeissa 111
Kostova, Blagovesta 339
Kotonya, Gerald 30
Kovar, Vladimir 258
Kuster, Christian 321
- Lamersdorf, Winfried 270
Lazovik, Alexander 174
Lenhard, Jörg 215
Leymann, Frank 379
Ludwig, Heiko 398
- Mann, Zoltán Ádám 296
Marir, Souad 111
Massot, Bertrand 93
Masuch, Nils 321
Megahed, Aly 188

- Metzger, Andreas 296
Mohamed, Mohamed 398
Murillo, Juan M. 124
Mutanu, Leah 30
- Nadgowda, Shripad 384
Nazeem, Ahmed 188
- Ortiz, Guadalupe 228, 369
- Palade, Andrei 149
Pallas, Frank 198, 240
Pautasso, Cesare 215
Pérez, David García 198
Pimentel, Ernesto 55, 363
Plebani, Pierluigi 198
Pratama, Azkario Rizky 174
- Ravindra, Pushkara 402
Reddy, Siva P. 402
Rosa-Gallardo, Daniel J. 228
Rößner, Ingo 252
Ruiz-Cortes, Antonio 352
- Saatkamp, Karoline 379
Sahli, Hamza 5
Sandobalin, Julio 346, 393
Sbaï, Zohra 80
- Schoenen, Stefan 296
Simmhan, Yogesh 402
Sivrikaya, Fikret 321
Stastny, Marek 258
- Tai, Stefan 198
Tata, Samir 188
- Vogel, Maximilian 283
Vukovic, Maja 384
- Wan, Kaiyu 137
Warke, Amit 398
Weber, Sebastian 283
Wehlitz, Robert 252
Werner, Sebastian 240
White, Gary 149
Widyawan 174
Willaerts, Bettina 18
Wolters, Dennis 388
- Yeddes, Moez 161
Yue, Yong 137
- Zavala, Edith 357
Zimmermann, Michael 379
Zirpins, Christian 283
Zrelli, Rym 161