Alexander Lazovik
Stefan Schulte (Eds.)

Communications in Computer and Information Science 707

Advances in Service-Oriented and Cloud Computing

Workshops of ESOCC 2016
Vienna, Austria, September 5–7, 2016
Revised Selected Papers

Springer

Communications in Computer and Information Science 707

Commenced Publication in 2007
Founding and Former Series Editors:
Alfredo Cuzzocrea, Xiaoyong Du, Orhun Kara, Ting Liu, Dominik Ślęzak, and Xiaokang Yang

Editorial Board

More information about this series at http://www.springer.com/series/7899

Alexander Lazovik · Stefan Schulte (Eds.)

Advances in Service-Oriented and Cloud Computing

Workshops of ESOCC 2016
Vienna, Austria, September 5–7, 2016
Revised Selected Papers

Springer

Editors
Alexander Lazovik
University of Groningen
Groningen
The Netherlands

Stefan Schulte
Vienna University of Technology
Vienna
Austria

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-319-72124-8 ISBN 978-3-319-72125-5 (eBook)
https://doi.org/10.1007/978-3-319-72125-5

Library of Congress Control Number: 2017962885

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the technical papers presented at the workshops collocated with ESOCC 2016 (the 5th European Conference on Service-Oriented and Cloud Computing), held in Vienna, Austria, during September 5–7, 2016. The workshops focused on specific topics in service-oriented and cloud computing-related domains and included the 4th International Workshop on Cloud for IoT (CLIoT 2016), the Second International Workshop on Cloud Adoption and Migration (CloudWays 2016), the Rethinking Services Research (ReSerCh) workshop, and the combined International Workshop on Performance and Conformance of Workflow Engines and International Workshop on Patterns and Pattern Languages for SOCC: Use and Discovery (PEACE in PATTWORLD). This volume also includes selected papers from the conference's PhD Symposium, and the papers presented at the European Projects Track (EU Projects 2016) in conjunction with ESOCC 2016.

The CLIoT 2016 workshop aimed at discussing the limits and/or advantages of existing cloud solutions for the Internet of Things (IoT) and at proposing original and innovative contributions for enhancing real-world resources over cloud environments. Smart connectivity with existing networks and context-aware computation is becoming indispensable for IoT. Cloud computing provides a very strategic virtual infrastructure that integrates monitoring devices, storage devices, analytics tools, virtualization platforms, and client delivery. It supports enormous amounts of data generated for IoT purposes, which have to be stored, processed, and presented in a seamless, efficient, and easily interpretable form. These features make the cloud computing a promising choice for supporting IoT services. IoT has the potential to offer the killer applications of cloud computing, where the cloud allows one to access IoT-based resources and capabilities, to process and manage IoT environments, and to deliver on-demand utility IoT services such as sensing/actuation as a service.

Regardless of the benefits of cloud computing, many organizations still rely on business-critical applications – legacy systems – developed over a long period of time using traditional development methods. In spite of maintainability issues, (on-premise) legacy systems are still crucial as they support core business processes. Therefore, migrating legacy systems toward cloud-based platforms allows organizations to leverage their existing systems deployed and provided (using publicly available resources) as scalable cloud services. CloudWays 2016 brought together cloud migration experts from both academia and industry. The workshop's goal was to promote discussions and collaboration among participants, to help disseminate novel cloud migration practices and solutions, and to identify future cloud migration challenges and dimensions.

Workflow management systems provide platforms for delivering complex service-oriented applications that need to satisfy enterprise-grade quality of service (QoS) requirements such as dependability and scalability. Benchmarking is an established practice that helps to drive continuous improvement of technology by setting a

clear standard and measuring and assessing its performance. Patterns have emerged in several IT domains as lingua franca to document proven solutions for frequently reoccurring problems. PEACE in PATTWORLD combines research in the field of performance and conformance of workflow engines with the field of patterns.

The fourth section of the proceedings includes selected papers from the ReSerCh workshop. Services have played a key role in innovating and revolutionizing most software-intensive realms from automotive to urban planning to business processes automation and so on. In a recent gathering of the IFIP working group on service-oriented computing (http://ifip-wg-sos.deib.polimi.it), it was evaluated that the sector is currently undergoing a deep change, which, if not correctly steered and encouraged, may leave service-oriented research in disarray. The goals of the ReSerCh workshop were twofold: On the one hand, it aimed to bring researchers and practitioners to share ideas in unconventional and innovative ways that harness service orientation for moving beyond current topics such as process automation or cloud computing. On the other hand, it aimed to distill the notion of services and service-oriented computing toward its originally intended tenets and challenges.

The main aim of the ESOCC 2016 PhD Symposium was to give PhD students an opportunity to present their research activities and perspectives, to critically discuss them with other PhD students and with established researchers in the area, and to get fruitful feedback and advice on their research activity. As for the main conference, the topics focused on all aspects of cloud computing, service-oriented architectures, Web services, and related fields. After the symposium, the students who presented results mature for a scientific publication were invited to submit an extended version of the presented paper. This post-symposium proceedings volume includes these papers.

Finally, the EU Projects 2016 track aimed at presenting the major running European-funded projects in the area of service-oriented and cloud computing, highlighting the main industrial and academic trends in terms of research and innovation.

The chairs would like to thank the individual workshop organizers, as well as all authors, keynote speakers, and participants. We also want to thank the main conference organizers for their support all along the process.

January 2018

Alexander Lazovik
Stefan Schulte

Organization

ESOCC 2016 was organized by TU Wien, Austria.

Organizing Committee

Workshop Chairs

Alexander Lazovik	University of Groningen, The Netherlands
Stefan Schulte	TU Wien, Austria

Workshop Organizers

Maria Fazio	University of Messina, Italy
Pooyan Jamshidi	Carnegie Mellon University, USA
Oliver Kopp	University of Stuttgart, Germany
eva Kühn	TU Wien, Austria
Joerg Lenhard	Karlstad University, Sweden
Frank Leymann	University of Stuttgart, Germany
Nabor C. Mendonça	University of Fortaleza, Brazil
Claus Pahl	Free University of Bozen-Bolzano, Italy
Cesare Pautasso	University of Lugano, Switzerland
Dana Petcu	West University of Temisoara, Romania
Pierluigi Plebani	Politecnico di Milano, Italy
Damian A. Tamburri	Politecnico di Milano, Italy
Guido Wirtz	University of Bamberg, Germany

EU Projects Track

Antonio Brogi	University of Pisa, Italy

PhD Symposium

Gianluigi Zavattaro	University of Bologna, Italy
Wolf Zimmermann	Martin Luther University of Halle-Wittenberg, Germany

Contents

CLoud for IoT

Preface of CLIoT 2016

Internet of Things (IoT) technologies are changing the way we interact with the world around us, and new business opportunities exploiting IoT solutions are arising. IoT conceptual base aims to represent the physical world through uniquely identifiable and interconnected objects (things). These things are able to capture information from the environment, process them and/or perform actuation within the real world, thus supporting the development of cyber-physical and autonomous systems in several applications domains. However, to achieve such challenging aims, the exploitation and orchestration of several heterogeneous physical and virtual resources is necessary. Cloud computing represents a very flexible technology, able to offer theoretically unlimited computing and storage capabilities, and efficient communication services. Cloud technologies address two important goals for distributed system: high scalability and high availability. These features make the Cloud Computing a promising choice for supporting IoT applications and services. However, appropriate Cloud solutions and strategies aimed at the IoT need to be investigated in order to verify effectiveness and efficiency. The *International Workshop on CLoud for IoT* (CLIoT) aims at bringing together scientists, practitioners and PhD students in order to discuss the limits and/or advantages of existing Cloud solutions for IoT, and to propose original and innovative contributions for enhancing real world resources over Cloud environments.

CLIoT 2016 is the fourth edition of the International Workshop on CLoud for IoT, and the topics of interest for CLIoT 2016 included but were not limited to:

- Innovative models and system architectures for Cloud based IoT
- IoT Data abstraction and processing
- Mobile Cloud
- Cloud storage for IoT
- Interaction between sensor networks and Cloud
- Discovery Service for IoT
- Cloud Computing based IoT technologies
- Wireless Sensor Networks into the Cloud
- Big data management using Clouds
- Smart Environments for IoT
- Ubiquitous computing/pervasive computing for IoT
- Real-time communication with smart objects
- Applications based on IoT and Cloud
- Inter-cloud management: Cloud Federation serving IoT
- Security and privacy in Clouds and IoT
- Edge, Fog and Dew Computing
- Micro-service architecture

All submitted manuscripts have been peer-reviewed by an international program committee, with the objective of having at least three reviews for each paper. The final acceptance rate of the manuscripts was 40%.

The contributions accepted for presentation at the workshop include the work of Celesti et al., which presented a Multi-Criteria Decision Making algorithm able to manage the migration of virtual machines among providers in order to lead towards global energy sustainability and cost-saving.

Cartier et al. overviewed the prevalent solutions and architecture design principles in IoT-big data ecosystems for smart cities sensing. Furthermore, they presented the needs of IoT-big data software ecosystems by exemplifying existing IoT systems.

García-Alonso et al. investigated the key challenges that must be faced to build distributed and context-aware human-centric Cyber-Physical Systems that take advantage of the capabilities of modern smart devices.

Giacobbe et al. presented the Big Bucket IoT Cloud environment, where smart dumpsters are equipped with low-cost sensors and open source easy-to-use hardware and software.

In the workshop program, also two short papers were included. Massonet et al. presented current work in the BEACON project to secure the federated network with a global security policy. Ahmadi Ghohandizi et al. discussed an application framework, development tool and execution platform for programmable remote devices.

Maria Fazio
Dana Petcu

Organization

Workshop Organizers

Maria Fazio	University of Messina, Italy
Dana Petcu	West University of Timisoara, Romania

Steering Committee

Nik Bessis	University of Derby, UK
Massimo Villari	University of Messina, Italy

Technical Program Committee

Liz Bacon	Greenwich University
Francisco J. Blaya Gonzálvez	University of Murcia, Spain
Antonio Celesti	University of Messina, Italy
Ciprian Dobre	University Politehnica of Bucharest, Romania
Andy Edmonds	Zurich University of Applied Sciences, Switzerland
Teodor-Florin Fortis	West University of Timisoara, Romania
Daniel Grosu	Wayne State University, USA
Manuele Kirsch Pinheiro	Université Paris 1 Panthéon Sorbonne, France
Natalia Kryvinska	University of Vienna, Austria
Brian Lee	Athlone IT, Ireland
Fei LI	Vienna University of Technology, Austria
Dan Marinescu	School of EECS University of Central Florida, USA
Tommi Mikkonen	Tampere University of Technology, Tampere, Finland
John Morrison	University College Cork, Ireland
Juan Manuel Murillo Rodríguez	University of Extremadura, Spain
Bogdan Nicolae	IBM Research, Ireland
Leire Orue-Echevarria	Tecnalia Research & Innovation, Spain
Chrysa Papagianni	National Technical University of Athens, Greece
Florin Pop	University Politehnica of Bucharest, Romania
Evangelos Pournaras	ETH Zurich, Switzerland
Luiz Angelo Steffenel	Université de Reims Champagne-Ardenne, France
Orazio Tomarchio	University of Catania, Italy
Jose Luis Vazquez-Poletti	Universidad Complutense de Madrid

IoT-Big Data Software Ecosystems for Smart Cities Sensing: Challenges, Open Issues, and Emerging Solutions

Alexander D. Cartier[1,2], David H. Lee[2], Burak Kantarci[2,3(✉)], and Luca Foschini[4]

[1] Assured Information Security, Inc., Rome, NY 13441, USA
[2] Clarkson University, Potsdam, NY 13676, USA
[3] University of Ottawa, Ottawa, ON K1N 6N5, Canada
burak.kantarci@uOttawa.ca
[4] University of Bologna, 40136 Bologna, Italy

Abstract. The Internet of Things (IoT) architecture primarily consists of massive amounts of heterogeneous objects, equipped with sensing, computing, and communication capabilities to continuously sense the smart cities pulse. The coordinated collection of this data produces relevant scalability and management issues not only in terms of communication but also in storage and computing to process and analyze large amounts of incoming big data streams. In these systems, people also play a pivotal role which includes both social and technical issues, making the design of these solutions a very complex task. This paper overviews the prevalent solutions and architecture design principles in IoT-big data ecosystems for smart cities sensing. Furthermore, we present the needs of IoT-big data software ecosystems by exemplifying existing IoT systems. We also provide useful insights towards future innovation to address open issues and challenges that are identified based on the expected growth of data in the next decade.

Keywords: Big data · Cyber-physical systems · Data analytics
Distributed computing · Fog computing · Internet of Things

1 Introduction

Motivated by the ever-increasing amount and value of data gathered and managed by Internet-born companies, such as Google, Facebook, and Amazon, big data software systems, specialized and tailored for the collection, storage, and analysis of those largest data depots, are already a well-established reality. Research efforts in this area not only highlights crucial issues, such as quality of collected data and scalability of the overall sparse and widely distributed big data systems, but also produced significant results, such as in the notable case of NoSQL database solutions (e.g., Cassandra and MongoDB) that, opposed to traditional SQL-based conventional database solutions, support horizontal

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 5–18, 2018.
https://doi.org/10.1007/978-3-319-72125-5_1

scaling by design [1,2]. Concurrently, advances in communications and device miniaturization enables the Internet of Things (IoT) vision, with smart objects in conjunction with smartphones – that in 2013 outnumbered normal feature phones – acting as sensors deployed over smart cities and equipped with sensing, computing, and communication capabilities.

At the current stage, several research activities in IoT software design focus on overcoming heterogeneity issues in both communication technologies and software Application Programming Interfaces (APIs) for data gathering and management (see also the IoT architectures and applications sidebar). The goal is to collect and analyze incoming big data flows, which are densely available and harvested in urban areas, allowing for a very large-scale fine-grained sensing, by exploiting all personal resources, mobile activities and collaborations. However, the software design principles to develop a new class of IoT-big data solutions defined as software ecosystems are still widely unexplored. For this to become an effective technology, IoT-big data systems for smart cities sensing still have to face several challenges. First, there has been a continuously growing network infrastructure associated with the rise of IoT, generating more data (i.e., high volume) at an exponentially increasing rate (i.e., high velocity). This data rate is overwhelming the conventional means of data management, calling for new methods of collecting, storing, and processing information collected from the Internet of Things. There are several exemplary models and technologies to implement these new concepts in big data management in accordance within the IoT framework. This ecosystem calls for an integrated architecture which combines the functions of Service-Controlled Networking (SCN) through the middleware, orchestrating the service components of the cloud [3]. From the social perspective, smart city sensing, which involves smartphone-enabled sensing, calls for the identification of willing users to participate in sensing campaigns, keeping them involved (e.g., by providing services, entertainment, and rewards), and fostering their participation with active collaboration in data collection. This requires user tasks to operate at specific locations (e.g., taking a picture of a monument, tagging a place, etc.). However, the boundary between social and technical challenges is not clear cut. The technical problem of minimizing the global resource overhead by entrusting a minimal subset of users in a sensing campaign requires analyzing their geo-social profile; to identify and infer which users are most likely to successfully harvest the required data [4].

Focusing on big data, this emerging trend has been clearly recognized by the market. IDC's Digital Universe study in 2012 reports 18% of the United States' digital universe is valuable (158 exabytes in 898 exabytes) when analyzed and tagged properly. In the same report, it is predicted that by 2020, the useful information will grow 17-fold corresponding to 40% of the digital universe forecasted for 2020 (i.e., 2631 exabytes in 6617 exabytes). Another important research discovery suggests that the cloud stored, processed, and transmitted 14% of the digital universe in 2012 whereas cloud-based services will host 37% of the digital universe in 2020. These numbers call for the adoption of big data solutions in IoT where volume and value of the data will be the driving factors in the evolution of

big data software, optimization for real time systems, cybersecurity and forensics for the next decade [5]. These numbers confirm that the challenges with the IoT-big data will keep evolving throughout the next several decades. This article aims to present a collection of methods, and theories, discuss prevalent architecture design principles and identify the required technologies by detecting open issues and challenges, which will provide insights for paving the way towards effective IoT-big data solutions. We provide a survey of remarkable big data solutions for the IoT, then we identify and discuss the open issues and challenges in the IoT-big data ecosystems.

2 IoT Architectures and Applications

The Internet of Things (IoT) pervasively and ubiquitously interconnects billions of devices with sensing, computing and communication capabilities as seen in Fig. 1.

Furthermore, it is crucial to collect, aggregate and correctly represent the data gathered at the sensor network level, where the data will be sent to the next level, namely the middleware. The middleware acts as a layer between the software application layer and hardware layer by parsing the data in order to recognize certain trends or specific patterns to create reusable solutions for frequently encountered problems such as heterogeneity, interoperability, security and dependability. It is worthwhile noting that the majority of middleware solutions currently do not provide the functionality of context-awareness and most solutions are focused on device management. Hence, it will be critical to continue to work on the implementation of context-awareness into the middleware solutions within the IoT [3].

In the IoT-big data ecosystem, the sensing plane consists of the sensors of various types; such as temperature, light, airflow, motion, humidity and several other sensors purposed for various applications including vehicular networks, water quality and e-health monitoring. The role of sensors in this architecture is continuously reporting the sensed data to the data plane through the middleware. The middleware, as mentioned previously is responsible for aggregating data from numerous sensors and presenting it to the data plane for pre-processing and storage. The data plane offers short and long term storage for the aggregated data, and it pre-processes the data for the Cloud platform which provides Data Analytics as a Service [6] where embedded analytics and statistics libraries play a key role [7]. It is worthwhile noting that the data plane can also be implemented within the cloud platform based on the storage-as-a-service concept in cloud systems. The application plane receives software-as-a-service (SaaS) from the cloud platform and interprets the analyzed data in accordance with the desired application e.g., e-health, smart metering, intelligent transportation systems, and so on.

In the corresponding architecture, data is collected via distributed sensors that are uniquely identifiable, localizable and communicable. The collected data goes from the user interaction with the embedded system, up to the local network level and is then stored either on local servers or in the cloud, at which point

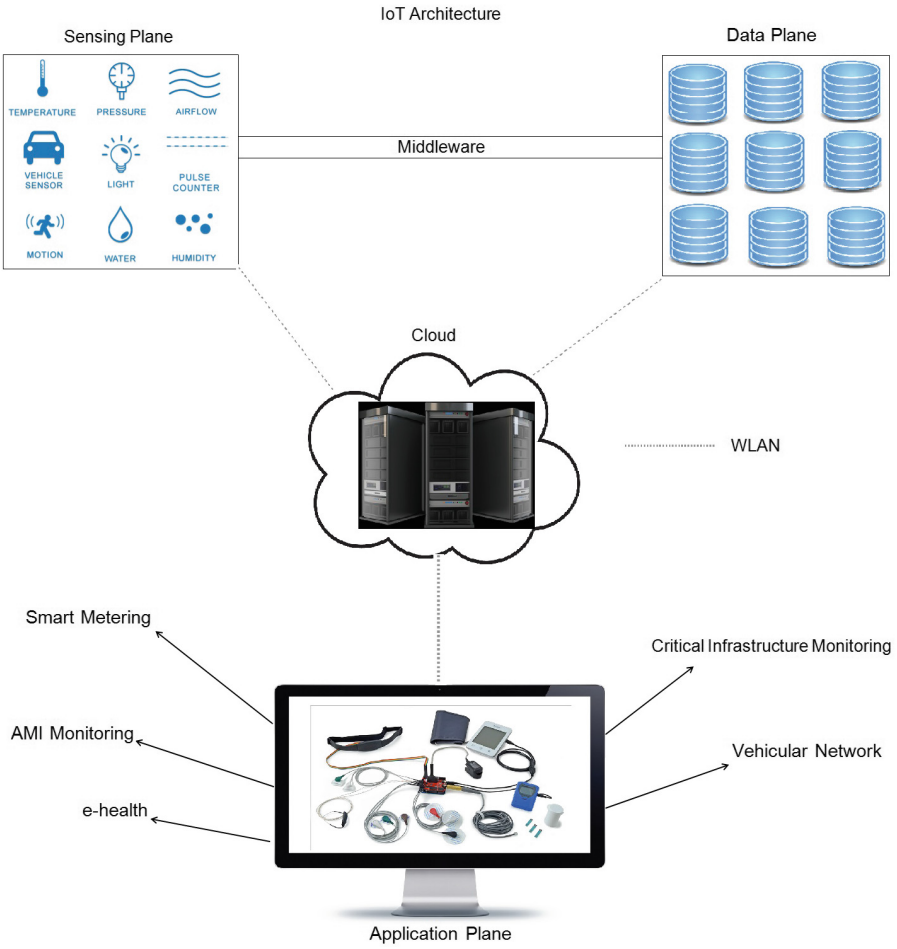


Fig. 1. The IoT-big data ecosystem with three planes that are connected through a cloud platform.

the data is available for a variety of uses. The IoT architecture interconnects sensors, RFID tags, smart phones, and other objects in a scalable manner.

In [8], the requirements of sensing objects driving the integration of cloud computing and IoT are summarized as having huge computing and storage capacity, web-based interfaces for data exchange and integration as well as programming platforms, real-time processing of big data, inter-operability between the sensing objects, cost-efficient, scalable on-demand access to the IT resources, and security and privacy assurance. Therefore, the authors propose deployment, development and management of the IoT applications over the cloud, namely the CloudThings architecture. Recent progress in IoT has not only been made in applications dealing with data analysis, but also in newfound approaches in

structure, storage, and compression. The common goal in the related works is to make the IoT data readily accessible and understandable to the end user.

A motivation of some of the recent progress has been a product of the Smart-Campus project that includes two different scenarios where sensors would be placed to determine the occupation rate of parking lots as well as regulating the temperature through the control of doors leading outside [9]. These services on the network would continuously collect data in real time in order to eventually recognize patterns. The middleware plays a key role in the implementation of this project because of several API's used to send data, set up the configurations for measurement retrieval, and to interact with collected data sets. The responsibility of the middleware is to support the data reception as well as broadcasting the configurations made on the sensors.

3 IoT-Big Data Design Guidelines

Figure 2 illustrates the need for big data management schemes in an IoT-dominated environment. Volume, variety and velocity of the data are driven by the data quality assurance needs, uncertainty in social media accounts, and networked devices being twice as the global population. Consequently, the traditional data warehousing solutions remain with low veracity in the IoT Era.

Starting from big data-related aspects, major trends in the field of big data gathering have an intense focus on the following four areas: velocity, variety, volume and veracity. Velocity denotes a focus on high-speed processing/analysis such as click-streaming and fast database transactions. Variety in the structures of data being collected arises along the lines of Machine-to-Machine (M2M), radio frequency identification (RFID), and different types of sensors. Volume includes currently used services such as social networks, cloud storage, network switches, thermo-metric/atmospheric/motion sensors, and so on. The primary issues that must be considered when focusing on these three subjects are the limitations of nodes' buffer sizes and the maximum acceptable latency in data collection [10]. Finally, veracity is defined as the potential of releasing useful information out of unstructured big data. Indeed, handling of the data through trusted sources improves the veracity of analytics as reported in [7].

The concept of big data has come about with a recent increase in the volume, velocity, variety, and veracity of data collected via various sources but mainly via IoT sensors. A cloud-based eco-system is envisioned to share and trade high-quality data from a vast network of independently managed sensors in real time [1,9]. While there has been considerable research with WSNs, using cloud-based platforms to host sensor networks is one of the biggest challenges yet, and the research regarding this topic has recently started.

This vision introduces a previously unexplored area of research. A few topics which must be addressed in order to find a solution to this challenge are focuses on high-quality data, an efficient collaborative emphasis for sharing/trading data, the need of a markup language that can not only handle the network but can also support data quality and enable domains to access live sensor feeds

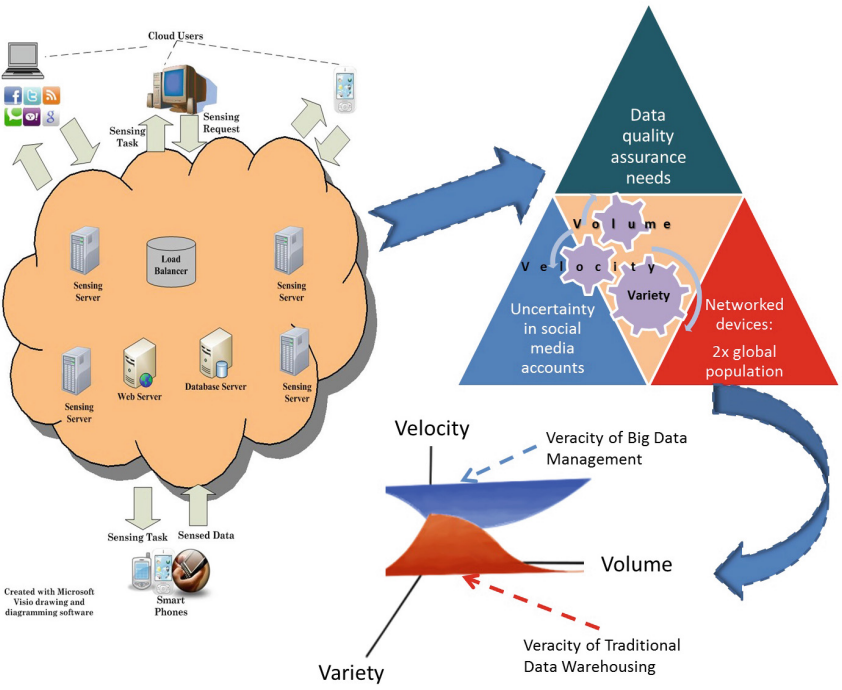


Fig. 2. The rise of the need for big data management in the IoT-dominated environment where majority of the data is collected by connected sensing devices.

as well as historical data [1]. According to this new vision, we propose some main design guidelines and concepts useful to compare existing solutions in IoT-big data literature (see also Table 1).

3.1 State of the Art in Building IoT-Big Data Architectures

Today, IoT-big data systems, such as data collected from the global Flightradar24 flight monitoring system, are handled via software chain architectures. The software chain basically maps processing phases of big data streams to multiple components denoting data generation, intermediate and result stages. As an example, the study in [11] uses the big data stream from a global flight monitoring system and is processed via the Yahoo!S4 framework. The Yahoo!S4 framework is mapped onto five stages, namely the sensor, extractor, parser, formatter and outputter modules.

The sensor module can be implemented as a script which captures unstructured data. The extractor module is responsible for identifying and distinguishing the events within the data streams. On each event, the parser module runs data analytics processes such as filtering, pattern recognition and data mining. The parser module can also be decomposed into multiple layers such as in

the Lambda architecture where Hadoop serves as the batch layer for long term data and Storm serves in the speed layer to manage real time data. Formatter and outputter are responsible for generation of the structured data out of the unstructured data under analysis and maintaining them in a file system of NoSQL database.

In an open IoT system, a similar software chain approach can be adopted. As today's technology is able to enable access to sensor readings through web-based services; the sensor component can obtain the data of the IoT sensors via APIs that enable access to web servers. The Open Geospatial Consortium's (OGC) IoT RESTful API has been built on the OGC Sensor Web Enablement standards in order to interconnect IoT objects, their data and applications over the web via JavaScript Object Notation (JSON) data interchange format. The API can be integrated to the sensor-end of the software chain in order to make various IoT sensors of multiple participants connect to the web servers that are compliant with the OGC standard [12].

3.2 Challenges Experienced in IoT-Big Data Systems

Focusing on IoT-big data systems, we identify some distilled guidelines based on experiences within the ParticipAct sensing project [4], and identify four major categories for design guidelines as support for spatio-temporal queries, minimal overhead on IoT nodes, openness and security, and fast feedback and minimal delay in producing quality-aware sensing data.

Support for Spatio-Temporal Queries. Support for spatio-temporal queries over sensed data is a key factor when considering big data because of all possible sources of where/when data might be pushed from, and it is also important to keep track of sensed data chunks for future uses. First of all, as data is neither temporally nor spatially static in the IoT, storage and scalability of retrieving the data appears as an important issue due to the constant movement of data.

Minimal Overhead on IoT Nodes. By this design guideline, we aim at minimizing energy consumption due to computing and communication at IoT nodes through optimizations of local sensing processes (such as, duty-cycling, employing physical models/verifications, etc.) and, most important, of sensed data transmissions toward the backend (such as, by locally bulking multiple data samples in the same sent packet, coordinating IoT nodes in the same location to avoid useless readings such as in WSN, etc.).

Openness and Security. Sensed data should be stored securely and encrypted to protect it from possible threats. This challenge has been tackled in [13] through the use of a distributed storage system using Shamir's secret sharing as the driving algorithm for both security and storage.

Fast Feedback and Minimal Delay in Producing Quality-Aware Sensing Data. This design guideline derives from the need to associate data with a quality indicator based on the history of data sensed in the past. This requires continuous profiling of sensed data in several different dimensions and grains (such as time, space, weather, season, etc.) by exploiting big data storage to keep all of these profiles ready, thus allowing fast computation for required feedback. Notable efforts within this direction are sensor webs such as IrisNet and SensorWeb. Furthermore, projects like Aurora, Borealis, Cayuga, Stanford Data Stream manager and System have explored many issues associated with stream and event processing comprising the construction of algorithms and techniques for data quality-aware sensor feed discovery service composition [1].

4 Remarkable Big Data Solutions

As big data continues to be researched, there has yet to be a single defining breakthrough when it comes to solutions regarding IoT. This is due to many variables that need to be considered when implementing an idea towards big data in the IoT such as volume, security and storage. While solutions are currently being researched and tested, there have been several instances of progress when dealing with this topic.

4.1 Crowdsensing-Based IoT-Big Data Projects

The features in IoT-big data design guidelines make ParticipAct a complete mobile crowdsourcing platform that encompasses the whole process from data collection, to post-processing, to mining, and is available to the mobile crowdsourcing community as an open-source project [4]. In a related study, after the detailed description of the whole architecture of ParticipAct and its technological stack, some of the use case scenarios are presented. The corresponding scenarios are currently being used to evaluate the potential of mobile crowdsourcing and ParticipAct both qualitatively and quantitatively.

4.2 Smart Environment Projects

The SmartCampus experiment performed on the SophiaTech campus in France [9]. The idea of this project focuses on the final product becoming an open platform for different types of campus members to use the already deployed sensors to build their own services or user defined sensors. Through this project, concepts such as data retrieval and user-defined sensors are implemented into realistic situations where big data and IoT are the focus. Data retrieval is applied in a way where users can pull sensor properties using input filters or just the sensor data itself. User-defined sensors are also introduced in this project as virtual sensors where users can define a specific configuration and store it into database where it can be executed using scripts when its dependencies produce data.

AllJoyn Lambda, a software architecture which integrates the Alljoyn framework into Lambda architecture to enable big data analytics for IoT applications [14]. The proposed architecture adopts the AllJoyn technology that is intended for IoT. However it aims at overcoming the real time processing/storage and management of the data obtained from smart environments by integrating the MongoDB NoSQL database for storage and Apache Storm for real time analysis of the data pushed from the smart objects.

As an emerging smart environment, the IoV concept has evolved from the IoT, and it is presented in [15] where nodes are represented by vehicles and are connected to form a Vehicular Ad Hoc Network (VANET). The biggest challenge in IoV is processing this volume of data and delivering it to its destination, which is done through various relay nodes. This paper analyzes the issue by using Bayesian Coalition Game (BCG) and Learning Automata (LA). The use of BGC trains the LAs to make moves correlating to each node/vehicle performing tasks to make each player in the game safer and more aware. This proposal adopts the Nash Equilibrium concept with respect to the probabilistic belief of players in the coalition game.

4.3 Edge Computing Based Projects

The Available Network Gateways in Edge Location Services (ANGELS) framework appeared as a result of the realization of the complexity of new applications in cyber-physical systems [16]. Services encompassing multiple domains are beginning to come into effect. As astronomical volumes of data collected will begin to require huge computing infrastructures for analysis, ANGELS introduces a framework for fog-computing, which utilizes a key aspect of the IoT field that has been overlooked thus far. The framework focuses on taking into account the ability of resources available prior to the distribution of tasks. Researchers have explored the idea of smart edge devices to perform portions of IoT data analysis where edge devices are low-powered computational nodes such as smart phones and home energy gateways. The proposed architecture consists of servers and commodity computing nodes as well as these smart edge devices as computational resources. This framework of heterogeneous computational nodes includes resources ranging from large server-class systems down to low-powered edge devices forming the basis of the fog computing paradigm. This solution also involves parallel data computation along with capacity based partitioning to accomplish a more streamlined approach to big data management.

4.4 Big Data Stream Analysis Projects for Cyberphysical Systems

The proposal in [17] is tailored for cyber-physical systems, and it presents an online spatiotemporal analysis, which would implement a grid-based single-linkage clustering algorithm over a sliding window. This online time-space efficient method satisfies the velocity demand of big data streams. A large-scale real-world scenario including 300,000 sensors over the course of a year has been established to evaluate the success of the algorithms.

The rising necessity for robust and reliable services is leading to the creation of enormous amount of data, which has the possibility of exceeding the storage capacity of current micro servers. This has led to Big data correlation orchestrator (BigCO) which was implemented in a micro cloud server [18]. In the same study, it is also addressed how multifaceted data could be interrelated and analyzed with 3D modeling. On top of that, a streaming algorithm that extends Ramer-Douglas-Peuker heuristic is presented. This proposed compression algorithm has achieved up to a 99.86% compression of sensor data. With the recognition of consistent growth in the number of wirelessly connected devices, the same study conducts in depth testing dealing with high volumes of data. Their compression method along with the 3D modeling of data assesses the velocity at which they can analyze large quantities of data from a varying pool of sensors. BigCO implementation on a micro cloud server also offers portability to the data collection and analysis mechanism. The overall design of this orchestrator exhibits high veracity throughout the compression, modeling, and the overall BigCO framework.

4.5 Distributed, Secure, Scalable Storage of IoT Data

In [13], a project focusing on secure and scalable IoT storage systems is presented where the security system is derived from Shamir's secret sharing algorithm. However, a major focus is also placed in terms of volume when referring to IoT storage systems. A distributed storage system was designed based on the idea of the algorithm where any sort of incoming data is transformed into scaled shares based on the size of the original files and is inaccessible without the retrieval of all shares. This method considers volume in relation to scalability. This is done through an infrastructure based on a client-peer system where a client takes incoming data; transforms them into scaled shares creating smaller data pieces which can eventually be reassembled to form the original file also taking security into consideration. In terms of performance, this system does not account for the velocity at which the data would be stored and retrieved due to a bottleneck.

4.6 Quality of Data (QoD)-Aware IoT Big Data Projects

In case of continuous retrieval of data through sensor feeds, it is important to focus on the quality of data being pulled down. The project in [1] breaks the idea of the Quality of Data (QoD) down into several aspects, which focuses on availability of sensor feeds, latency, and trustworthiness. These qualities can be used to determine the certain attributes of managing big data in a system where attributes such as trustworthiness of data can be defined as veracity of data or accuracy. Other attributes, such as latency and availability of sensors correlate with the velocity of data being pulled down. The idea behind this project was to create a model for sensor services, where, in order to enable seamless sharing of sensor feeds from various sensors coming from different sources through the cloud. While QoD takes into consideration velocity and veracity of the sensor feeds, they stray from other aspects of big data such as volume and variety.

Though a major focus is placed on the variety of data through the use of heterogeneous sensor feeds, this work does not fully aim to address the variety aspect of big data.

4.7 Spatial Big Data Projects

Due to a gap in the development and applications of integrated information systems for snowmelt flood early warning in water resource management, an integrated system with IoT, geo-informatics (GIS, GPS, etc.), and cloud services have been proposed for the monitoring and simulation of snowmelt flooding [19]. This study resulted in an increase in the effectiveness of decision-making because of the availability of data and the integrated system to analyze all of the data in an efficient enough manner to make a difference when it comes to split-second decisions. This proposal goes off a popular practice in the field, which utilizes environmental tracking and analysis. The architecture of this system collects a wide variety of data from an assortment of information acquisition facilities. The collected data calls for a storage facility that can handle large volumes of information which this system architecture also takes into account. Then the proposals computing and analysis facility as well as the network and software used in this system accounts for high velocity collection and analysis of the data without a loss in veracity through the whole process.

Projects specifically addressing spatial big data exhibits new challenges where performance on large amounts of measurements is associated to specific locations and the instance of time when they were conducted. In addition, the paradigm of wireless networks data analytics varies from the classical data-mining paradigm as it poses different challenges in scalability and computational time. While relational data requires a linear time scale for computation for classification and prediction, spatial data requires a cubic time scale which causes problems for scalability both in terms of volume, calculation period and velocity. Although it remains as a distinct category from relational big data, the research in [20] has shown that existing parallel processing and computational framework algorithms are powerful tools for implementing spatial processing frameworks but the proper architecture for these tools are still being researched.

Dealing with spatial big data is considered one of the key challenges for the development of future wireless networking applications in terms of big data. Some underlying issues with this topic are that it currently requires a high level of specialized knowledge in order to design and implement systems for processing spatial big data. In order for this technology to grow and expand, it requires a wider use of context and development of systems such that non-experts in the field are able to build various applications using this technology. Therefore, a more common language for reasoning and computational inference solutions is necessary for development of these systems.

Table 1. Summary and comparison of the surveyed solutions

Application	Volume	Velocity	Variety	Veracity	Spatio-temp. queries	Min. overhead	Openness-security	Quality-aware sensing
Participact [4]	√	√	√	√	√	√	√	√
Smart campus [9]	√	√	√	×	√	×	×	×
Alljoyn-Lambda [14]	√	√	√	×	√	×	×	√
IoV [15]	√	×	×	√	×	√	×	√
ANGELS [16]	√	×	√	×	×	√	×	√
CPS [17]	√	√	×	×	√	√	×	×
BigCo [18]	√	√	√	√	×	√	×	×
Dist. storage [13]	√	×	×	×	×	×	√	×
Quality of data [1]	×	√	×	√	×	×	√	√
Flood warning [19]	√	√	√	√	√	√	×	×
Spatial data [20]	√	√	×	×	√	×	×	√

5 Summary

This paper has introduced the challenges in the IoT-big data ecosystem, and overviewed recent applications as summarized in Table 1. A majority of these applications focuses on two or three of the four Vs typical of big data systems, yet solutions addressing all dimensions are emergent. Veracity is the most neglected dimension in related work; hence trustworthiness assessment modules are emergent in software architectures that are proposed for IoT data management. Focusing on specific IoT-big data challenges, availability of tools and libraries for embedded data analytics are critical for use in the development of middleware solutions for the IoT-big data ecosystem.

All these challenges, pointed out by the related work, will impact the handling of the IoT data which is expected to contribute to the majority of the data accumulation in the near future. NoSQL-based solutions are feasible to overcome the storage challenges of big volumes of data while realtime analytics solutions such as Storm or Spark suit well with the IoT stream data of high velocity.

6 Open Issues and Challenges

As the data will be scaled out to higher volumes, varieties and velocities with the wide adoption of connected devices, IoT and big data will be two inseparable phenomena of the future. Major innovation should be on the analytics software architecture that can handle analytics on long term and real time data. Despite the availability of architectures, like Lambda, that address this issue, optimization software is required for real time systems. OGC's standardization efforts for accessing IoT data is invaluable as heterogeneity of big data pushed by the IoT objects can be handled. More importantly, we expect that even by 2020 less than half of the digital universe will consist of useful data because the majority of the data will still be unstructured and untagged. Therefore, collection of the data, proper tagging and structuring to improve the value of the IoT

data is critical. Furthermore, how to secure the data collected from IoT devices and ensure privacy of personalized devices is a key problem. In fact, cloud-based storage, processing and transmission of the data will be reaching 40% of the digital universe, introducing security as a service in cloud analytics as an emergent issue. Finally, IoT sensors will push IoT data continuously, and typically in raw form to be processed so to produce value. Hence, development of scalable and analytics-backed visualization mechanisms for long term data is also important to prevent data overloads for the IoT-big data systems.

Acknowledgments. This material is based upon work supported by the U.S. National Science Foundation (NSF) under Grant No. CNS1464273.

References

1. Ramaswamy, L., Lawson, V., Gogineni, S.V.: Towards a quality-centric big data architecture for federated sensor services. In: IEEE International Congress on Big Data, pp. 86–93 (2013)
2. Gorton, I., Klein, J.: Distribution, data, deployment: software architecture convergence in big data systems. IEEE Softw. **32**(3), 78–85 (2015)
3. Sowe, S., Dong, M., Kimata, T., Zettsu, K.: Managing heterogeneous sensor data on a big data platform: IoT services for data-intensive science. In: IEEE 38th International Computers, Software and Applications Conference Workshops, pp. 295–300 (2014)
4. Cardone, G., Corradi, A., Foschini, L., Ianniello, R.: ParticipAct: a large-scale crowdsensing platform. IEEE Trans. Emerg. Top. Comput. **4**(1), 21–32 (2015)
5. Gantz, J., Reinsel, D: IDC: the digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east (2012). https://www.emc.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf
6. Talia, D.: Clouds for scalable big data analytics. IEEE Comput. **46**(5), 98–101 (2013)
7. Louridas, P., Ebert, C.: Embedded analytics and statistics for big data. IEEE Softw. **30**(6), 33–39 (2013)
8. Zhou, J., Leppanen, T, Harjula, E., Ylianttila, M., Ojala, T., Yu C., Jin, H., Yang, L.T.: CloudThings: a common architecture for integrating the Internet of Things with cloud computing. In: IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 651–657 (2013)
9. Cecchinel, C., Jimenez, M., Mosser, S., Riveill, M.: An architecture to support the collection of big data in the Internet of Things. In: IEEE World Congress on Services (SERVICES), pp. 442–449 (2014)
10. Takashi, D., Nishiyama, H., Kato, N., Miura, R.: Toward energy efficient big data gathering in densely distributed sensor networks. IEEE Trans. Emerg. Top. Comput. **2**(3), 388–397 (2014)
11. Xhafa, F., Naranjo, V., Caballe, S., Barolli, L.: A software chain approach to big data stream processing and analytics. In: Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), pp. 179–186 (2015)
12. The OGC (Open Geospatial Consortium) SensorThings API Standard. https://ogc-iot.github.io/ogc-iot-api/index.html
13. Jiang, H., Chen, S., Li, K.-C., Jeong, Y.-S.: A secure and scalable storage system for aggregate data in IoT. Future Gener. Comput. Syst. **49**, 133–141 (2015)

14. Villari, M., Celesti, A., Fazio, M., Puliafito, A.: Alljoyn Lambda: an architecture for the management of smart environments in IoT. In: International Conference on Smart Computing Workshops, pp. 9–14 (2014)
15. Kumar, N., Misra, S., Rodrigues, J., Obaidat, M.: Coalition games for spatio-temporal big data in internet of vehicles environment: a comparative analysis. IEEE IoT J. **2**(4), 310–320 (2015)
16. Mukherjee, A., Paul, H.S., Dey, S., Banerjee, A.: Angels for distributed analytics in IoT. In: IEEE World Forum on Internet of Things (WF-IoT), pp. 565–570 (2014)
17. Fu, Z., Almgren, M., Landsiedel, O., Papatriantafilou, M.: Online temporal-spatial analysis for detection of critical events in cyber-physical systems. In: IEEE International Conference on Big Data, pp. 129–134 (2014)
18. Mozumdar, M., Shahbazian, A., Ton, N-Q.: A big data correlation orchestrator for Internet of Things. In: IEEE World Forum on Internet of Things (WF-IoT), pp. 304–308 (2014)
19. Fang, S., Xu, L., Zhu, Y., Liu, Y., Liu, Z., Pei, H., Yan, J., Zhang, H.: An integrated information system for snowmelt flood early-warning based on Internet of Things. Inf. Syst. Front. **17**(2), 321–335 (2015)
20. Jardak, C., Mahonen, P., Riihijarvi, J.: Spatial big data and wireless networks: experiences, applications, and research challenges. IEEE Netw. **28**(4), 26–31 (2014)

Re-powering Service Provisioning in Federated Cloud Ecosystems: An Algorithm Combining Energy Sustainability and Cost-Saving Strategies

Maurizio Giacobbe, Antonio Celesti, Maria Fazio(✉), Massimo Villari, and Antonio Puliafito

Department of Engineering, University of Messina,
Contrada Di Dio (S. Agata), 98166 Messina, Italy
{mgiacobbe,acelesti,mfazio,mvillari,apuliafito}@unime.it

Abstract. Cloud federation offers new business models to enforce more flexible energy management strategies. Independent Cloud providers are exclusively bounded to the specific energy supplier powering its Data Centers. The situation radically change if we consider a federation of cooperating Cloud providers. In such a context a proper migration of virtual machines among providers can lead to global energy sustainability and cost-saving strategy. In this paper, we discuss a decision system for Cloud federation brokerage able to combine these two strategies. More specifically, we present Multi-Criteria Decision Making (MCDM) algorithm able to discover the most convenient Cloud providers candidate to join a particular energy-aware federation. In the end, modelling different possible real Cloud providers, we demonstrate how the algorithm can accommodate different Cloud federation scenarios characterized by particular energy-aware parameters.

Keywords: Cloud computing · Federation · Brokerage
Energy management · Energy efficiency · Energy sustainability
Energy cost-saving

1 Introduction

Population Reference Bureau (PRB)'s projections in the 2015 Data Sheet [1] show world population reaching 9.8 billion by 2050, up from an estimated 7.3 billion now, with education (i.e., percentage quota enrolled in Secondary School) greater than 76%. This trend will produce new needs and desires by people, that will cause an increase of energy demand to access digital technologies. As a consequence, it is required to re-powering infrastructures, platforms and services by adopting sustainable and cost-saving solutions.

The European Commission's Effort Sharing Decision [2] forms part of a set of policies and measures on climate change and energy (i.e., the climate and energy package) that will help move Europe towards a *low-carbon economy*, moreover

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 19–33, 2018.
https://doi.org/10.1007/978-3-319-72125-5_2

increasing European energy security. The ESD establishes binding annual greenhouse gas emission targets for EU member states for the period 2013–2020. Furthermore, *Digital Single Market (DSM)* is one of ten European political priorities because EU, at the same time, is considering to overcome the national barriers of each of its member state, by moving from its internal national markets to a single one, and to overcome the current jungle of ICT standards [3].

In this context, Cloud computing and new smart digital technologies promise to transform our everyday life, towards a *new digital age*. In particular, Cloud Federation is an emerging topic that allows to carry out new business opportunities for Cloud service providers in many application field, including the energy management. Independent Cloud providers are exclusively bounded to the specific energy supplier powering its Data Centers. The situation radically change if we consider a federation of cooperating Cloud providers. We define a Cloud Federation as a mesh of Cloud providers that are interconnected based on open standards to provide a universal decentralized computing environment where everything is driven by constraints and agreements in a ubiquitous, multi-provider infrastructure [4]. In such a context a proper migration of virtual machines among providers can lead to different energy management strategies.

In this paper, we discuss a strategy based on a Multi Criteria Decision Making (MCDM) approach that allows a Cloud broker to combine energy sustainability and cost-saving factors along with service parameters in order to detect the most affordable destination federated Cloud provider where services can be migrated. Our strategy can be adopted in different brokerage schemes to determine how partnerships should be established in a Federation so as to allow the whole federated Cloud ecosystem to pursue a global energy management strategy, e.g., pushing down energy costs and/or greenhouse emissions (e.g., carbon dioxide).

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 motivates our work. Section 4 presents the main parameters that have to be considered to plan ahead our strategy. Section 5 presents our multi-criteria decision algorithm for an energy-aware broker in a federated Cloud environment. In Sect. 6, we presents our analytic evaluations considering real parameters, thus proving the goodness of our approach. Section 7 concludes the paper.

2 Related Work

Scientific literature presents several contributions on green Cloud and the intra-Datacenters energy management, but the most of the energy-aware management strategies are focused on independent Cloud datacenters. In [5] a review of literature on Cloud Brokerage Services is presented. However, even if the authors inspect several Cloud broker models, Cloud Federation is not a surveyed topic. The Reservoir European research initiative [6] explores the notion of a federated Cloud in which computing infrastructure providers lease excess capacity to others in need of temporary additional resources. The proposed sharing and paying model helps individual providers avoid over-provisioning of resources to deal with spikes in capacity demand. Hamze et al. [7] present a very interesting framework which addresses resource allocation according to an end-to-end

Service Level Agreement (SLA) established between a Cloud Service User (CSU) and several Cloud Service Providers (CSPs) in a Cloud networking environment. Compared to that study, our work is mainly focused on the achievement of a good compromise between sustainability and cost metrics, taking into account both mandatory and recommended criteria.

The authors of [8] propose a work based on a multi-criteria optimization technique for better selection of a service provider. They use Pareto front to decide the Cloud service provider which satisfies the *Quality of Service (QoS)* requirements of the user. This work focuses only on the QoS parameters (i.e., throughput and response time) but it is far from our purposes because it does not cover the dynamic composition of services based on the migration of data. *Software as a Service (SaaS)* adoption presents serious and unique security risks. In [9] the authors propose a holistic model, known as the *Function, Auditability, Governability and Interoperability (FAGI)*, as an approach to help a Cloud service consumer to engage and select a trusted Cloud service provider through four major decisions. The main goal of ECO2Clouds European project [10] is on assessing and reducing the environmental impact of Cloud applications. In ECO2Clouds, the problem of energy consumption and carbon dioxide emissions is addressed at the software level, assuming that optimization at the server and operating system level is not under control of the Cloud platform or the users. Compared to the ECO2Clouds scientific contributions, such as [11], our approach mainly differs for the brokerage role in our energy-aware strategy at Federation. In [12], the authors present a survey that helps researchers to identify the future trends of energy management in Cloud Federation. In particular, they select the major contributions dealing with energy sustainability and cost-saving strategies aimed at Cloud computing and Federation.

3 Motivation

In order to realize a cooperative ecosystem where federated Cloud providers can share their ICT resources to achieve energy sustainability and cost-saving objectives, a suitable large scale strategy is needed. It should include requirements and usage scenarios to use Cloud computing in both public sector and industry. More specifically, we mainly identify two typologies of requirements: mandatory and recommended. With mandatory we intend a requirement that an institution or authority, public or private, must absolutely have in order to provide a service. For example, public authority can demand to Cloud service providers to store data in Data Centers (DCs) that are localized in Europe. With recommended, instead, we intend a requirement that offers good benefits to an institution or authority, public or private. For example, an European public authority can recommend a price for a Cloud based service. A federated Cloud approach enables multiple independent Cloud providers to maximize their mutual benefit and therefore their business. It is able to encourage new investments in innovation thus to create a more competitive digital market. Moreover, a Cloud Federation can allow both small/medium-size enterprises and new companies

to become Cloud service providers, in according to specific requirements (e.g., *Service Level Agreements (SLAs)*, interoperability, certification, legislation). In such a context, and independently from the purposes that induce a Cloud provider to establish a Federation relationship with other providers (e.g., resilient scalability of resources, deployment of distributed services, energy sustainability, cost reduction, etc.), *brokerage* plays a fundamental role. We define brokerage as the process that allows a provider to select one or more external providers with which establish a federation relationship, on the basis of specific requirements. To this end, different possible schemes exist. In this paper, as Fig. 1, we consider the following schemes: **(i) centralized**, i.e., a single broker in charge of establishing the Federation; **(ii) hierarchical**, i.e., a set of brokers interacting each others to establish the Federation; Independently from the adopted brokerage scheme, the aim of this paper is to propose an algorithm for a Cloud brokerage system enabling federated Cloud DCs to enforce a dynamic energy management strategy. This is possible by performing a smart VM migration among federated Cloud DCs considering the best energy cost effective destination. Our reference scenario is not static, because the energy efficiency of a DC can change in given moments of the day and in given periods of the year. In fact, the cost due to energy in a given geographical area can change according to the time zone.

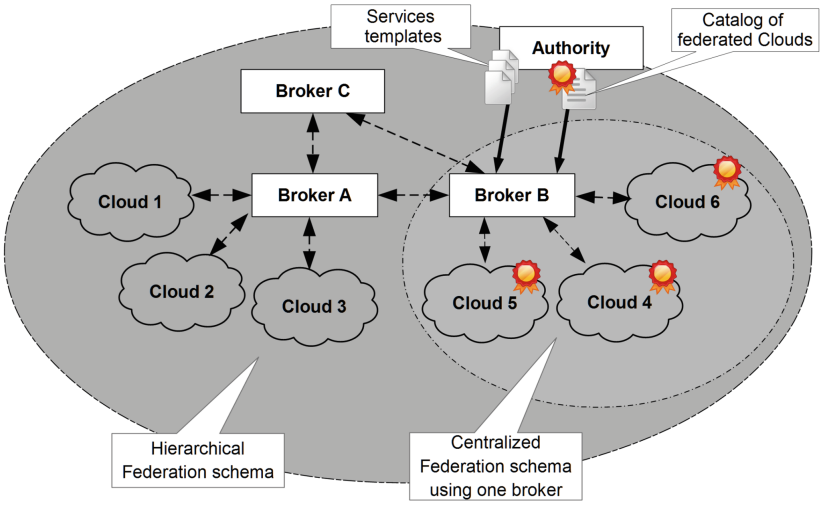


Fig. 1. Cloud brokerage schemes.

4 Strategy

Figure 2 shows our two-steps energy-aware strategy, based on a centralized Cloud Federation.

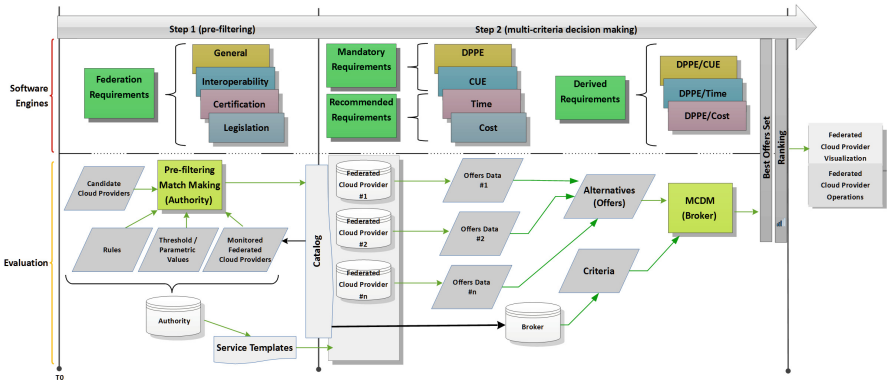


Fig. 2. The two-steps Energy-aware Strategy.

4.1 Pre-filtering and Matching Steps

As a first step, it includes a *pre-filtering* phase of the *federable* Cloud providers on the basis of several *federation requirements* (e.g., interoperability, certification, legislation, and other possible general requirements). If a Cloud provider fulfills all federation requirements, it is recognized by the authority in charge (i.e., a legitimate public authority such as the European authority for *Digital Agenda* in the Europe 2020 strategy), thus to became *federated* member of an inclusive e-society and to operate in an *energy-aware* DSM according to the possession of the above mentioned specific federation requirements.

To make this pre-filtering phase, and thereafter for the periodic monitoring on the possession of the federation requirements by the federated members, the authority is able to dynamically update, by *software engines* a *catalog* of all federated members. Each one of these is uniquely identified by an ID (for example we can think to use the tax code or similar), and the compliance with federation requirements is verified on the relative indicators (i.e., rules, thresholds or parametric values) established by the authority. In this manner the authority is able to control that all the Federation requirements are always satisfied and to provide in real time the same catalog to the brokers.

Therefore, in a second step our strategy provides for dynamically matching requests with offers at broker. More specifically, a broker is able to select one or more offers from one or more federated Cloud providers (identified through the catalog) on the basis of both mandatory and recommended energy and service related requirements as discussed in Sect. 3. In particular, an applicant federated Cloud provider, we name *Source*, dynamically makes its request to the broker by filling up in real time a *service template*. It is established and supplied by the authority in charge to both Cloud broker and service providers. We define a service template as a digital model which can be dynamically updated by an authorized entity (e.g., the broker, authorized provider, but also a public administration or a company in a more wide scenario).

For example, a Source can ask for resources to deploy scalable services or run microservices (e.g., Docker), data integration and business analytics platform (e.g., Pentaho), software for reliable, scalable, distributed computing (e.g., Apache Hadoop). For our energy-aware purposes, a Source uses a service template as a dataset (e.g., a vector or array) of requirements that will be useful to define a particular *workload footprint* thus to identify the environmental impact which is produced to run that workload at a Cloud provider.

4.2 Mandatory Requirements

Generally, the ecological footprint calculation is a set of complex operations that need to take into account several variables. Furthermore, the most famous metrics to measure energy efficiency of a Cloud Data Center, i.e., *Data Center infrastructure Efficiency (DCiE)* and *Power Usage Efficiency (PUE)*, are not sufficient because they do not take into account dynamics linked to the type of service offered. In order to overcome the above barriers, thus computing the workload footprint which is product at each possible destination and for each offer, we mainly identify the following mandatory fields: **DPPE** and **CUE**.

The Data Center Performance per Energy (DPPE) Metric. DPPE is a sustainability metric introduced by the *Japan's Green IT Promotion Council* in order to improve on PUE. It is defined as follows:

$$DPPE = ITEU * ITEE * 1/PUE * 1/(1 - GEC) \tag{1}$$

where DPPE has four sub-metrics, taking contemporary into account the necessity in to improve the Information Technology (IT) Equipment Utilization (ITEU), the IT Equipment Efficiency (ITEE), the efficiency of facilities with PUE, and purchasing 'green' energy with Green Energy Coefficient (GEC). Let's assume that each offer is associated with a site, and that the provider dynamically updates the DPPE of a site measuring in real time the relative sub-metrics (without exposing functionalities deemed sensitive or risky for its own business). DPPE characterizes the offer, and a greater value in DPPE indicates a greater energy efficiency.

The Carbon Usage Effectiveness (CUE) Metric. In order to address carbon emissions associated with DCs, the *Green Grid consortium* proposes the Carbon Usage Effectiveness (CUE) metric. By considering CUE, our strategy takes into account the impact of operational carbon usage, because it is mainly focused on correlating economical advantageous offer with a low-carbon Cloud site. CUE is defined as follows:

$$CUE = CEF * PUE \tag{2}$$

where CEF stand for *Carbon Emission Factor*, i.e. $kgCO_{2eq}/kWh$ of the site. It depends from the region where the Cloud site is located and it is based on the government's published data for that region of operation for that year.

4.3 Time and Cost as Recommended Requirements

In order to finalize the economic offer, we also consider as recommended fields the following parameters:

- δt, i.e., a normalized time index referred to the computational plus data transfer time;
- **cost**, i.e., the economic offer price.

The δt index is a normalized value with respect to a reference time for a specific type of request. For each type of possible request the reference template, that is made accessible by the authority in charge, contains a recommended field that represents the above-mentioned reference time. Cost can be expressed as a money/service unit ratio, for example in $/h (i.e., U.S. dollar/hour), or in $/GB (i.e., U.S. dollar/GigaByte).

4.4 Derived Requirements

The broker automatically computes the following ratios to complete each offer:

- **DPPE**/δt, i.e., how much *'green'* is the offer in relation to δt;
- **DPPE/cost**, i.e., how much *'green'* is the offer in relation to the economic offer price;

Our strategy, in fact, stipulates that it is the Cloud broker, and not each provider, the responsible to complete each offer in order to reduce data transfer costs and run the final multi-criteria decision step. In this manner, a Cloud broker is able to have in real time a knowledge of all requests and offers for several typologies of services, thus to match in real-time a set of offers for each request.

5 Multi Criteria Decision Making Applied to Energy-Aware Management in a Federated Cloud Ecosystem

On the basis of the above discussed aspects, the Cloud broker collects in real time all the energy and service related offers (codified in quantifiable form) from all federated Cloud providers. The main objective of the energy-aware Cloud broker decisional system is to pick out a set of offers that meets specific requirements. To this end, our strategy implements a Multi Criteria Decision Making (MCDM) algorithm that has been adapted to address the requirements of energy-aware federated Cloud environments using a *multi-criteria* matrix. In this Section, we present MCDM and discuss how it is adopted to design the multi-criteria decisional system.

The MCDM algorithm allows an energy Cloud broker to solve a decisional problem in which, according to M alternatives and N decisional criteria, we have to identify the best alternative or a set of Y alternatives so that $2 \leq Y \leq M$.

We define the set of alternatives $A = \{A_1, A_2, ..., A_M\}$ and the set of decisional criteria $C = \{C_1, C_2, ..., C_N\}$. For each Cloud provider that matches the requirement of the energy-aware federation, an entry including its quantifiable parameter is created inside the *MxN-dimensional* D matrix, as represented in Fig. 3.

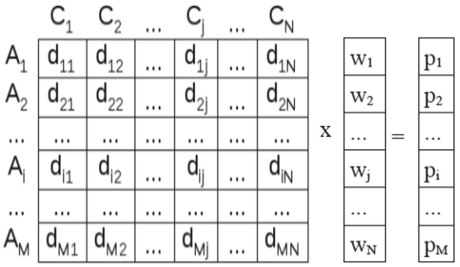


Fig. 3. Multi-criteria matrix multiplication.

The i-th row of the multi-criteria D matrix is a vector A^{i-th}, whose N elements are the quantifiable parameters of all criteria corresponding to the i-th economic offer (i.e., the i-th alternative) of a Cloud provider, whereas the j-th column represents a vector C^{i-th}, whose M elements are the quantifiable parameters of all Cloud providers corresponding to the j-th criterion.

The main operative steps in using MCDM algorithm in our energy-aware strategy include:

1. Identification of the energy-aware decisional criteria. More specifically, in our strategy we consider a set of seven decisional criteria $C = \{DPPE, CUE, \delta t, cost, DPPE/CUE, DPPE/\delta t, DPPE/cost\}$.
2. Identification of the *MxN-dimensional* D multi-criteria matrix, whose d_{ij} elements represent the measure of the alternative A_i on the criterion C_j;
3. Association of numerical measures to criteria, according to the impact that they have on the alternatives by means of the identification of the weight vector $W = w_1, w_2, ..., w_N$.
4. Processing of the numerical values with the Weighthed Sum Model (WSM) method consisting of the multiplication of the D multi-criteria matrix and the W vector. This results in the global evaluation vector $P = DxW$, with $P = p_1, p_2, ..., p_M$ as shown in Fig. 3, whose generic p_i element represent the global i-th alternative (i.e., the evaluation of the i-th alternative considering all decision criteria).

The i-th element of P is given by

$$p_i = \sum_{j=1}^{N} x_{ij} \cdot w_j \; with \; i = 1, 2, ..., M;$$

5. Identification of the best alternative or a set of Y alternatives so that $2 \leq Y \leq M$. The A_i alternative (i.e., the i-th Cloud provider offer) will better than the A_k alternative (i.e., the k-th Cloud provider offer) if and only if the corresponding global evaluation $p_i \geq p_k$, that is $A_i \geq A_k \Leftrightarrow p_i \geq p_k$ with i and k so that $i = 1, 2, ..., M$, $k = 1, 2, ..., M$, $i \neq k$.

The algorithm allows a dynamic evolution for an energy-aware Cloud federation. In particular, it enables energy-aware Cloud brokers to address energy sustainability contemporary taking into account other heterogeneous requirements identified as criteria in the matrix.

6 Analytic Evaluation

In order to evaluate our MCDM algorithm, we set a simulated scenario by using *Scilab* [13], a free and open source software for numerical computation. It provides a powerful computing environment for engineering and scientific applications.

6.1 Scenario

We consider three typical scenarios for our simulations, with respectively a number of three, ten and thirty federated Cloud Service Providers (CSPs). Each CSP is able to formulate from one to three offers (i.e., alternatives in our MCDM algorithm). On this basis, for each scenario we run our simulations to compute the global evaluation vector P (whose definition has been introduced in Sect. 5). More specifically, as a first step, we set the simulated environment starting from real datasets: (i) the sustainability sub-metrics dataset (Table 1a) and (ii) the time-cost fields dataset (Table 1b). They respectively concern: (i) the sub-metrics used to calculate the mandatory DPPE requirement; (ii) the recommended time-cost decisional criteria in order to finalize the economic offer.

The sub-metrics dataset is based on the measurement results of a real scenario, i.e. the *METI project* [14] that clearly shows the characteristics and the energy efficiency in several Japanese and Asian DCs, proving that the DPPE

Table 1. Datasets

sub-metrics	range
ITEU	0.3 - 0.6
ITEE	0.1 - 3.9
PUE	1.4 - 2.3
GEC	0 - 0.003
CEF	0.001 - 0.5

(a) Sustainability sub-metrics dataset.

recommended fields	range
δt	1.0 - 3.0
cost [$/h]	0.10 - 0.25

(b) Time and cost fields dataset.

is a useful assessment tool. In Table 1b, the recommended time requirement δt (which is previously introduced in paragraph Sect. 4.3) is a sheer number in the range from one to three times the reference time for a specific type of request. The *cost* range, instead, is defined on the basis of typical service prices of known worldwide providers (e.g., Amazon Web Services (AWS)).

Once criteria have been defined, we set the weight for each criterion. Specifically, we choose the same weight for each one of them according to have the same impact on the alternatives. This step results in the identification of the weight vector W in our algorithm.

6.2 Simulation and Experimental Results

This paragraph reports the results produced by the simulations, each one based on a number of 1000 samples. A graphical representation allows the reader for a quick visual feedback. Results are mapped on the basis of the derived requirements introduced in Sect. 4 and shown in Fig. 2. Therefore, it is possible to distinguish three scenarios (for 3-10-30 CSPs) and three typologies of evaluation map combining the derived requirements. Each evaluation map identifies "where" each CSP' offer is placed taking into account both the number of federated CSPs and a specific combination of derived requirements (i.e., the x and y-axes). More specifically, simulations place offers in three different typologies of map, that are the *DPPE/cost - DPPE/*δt for the *evaluation map 1*, the *DPPE/CUE - DPPE/*δt for the *evaluation map 2*, and the *DPPE/cost - DPPE/CUE* for the *evaluation map 3*. The above typologies are properly chosen taking into account the necessity in to reduce both energy consumption and carbon dioxide emissions, according the requirements related to the business of each provider (i.e., taking into account both energy-aware and time/costs service requirements).

The chosen graphical representation in Figs. 4, 5 and 6 allows to have both a quantitative (number of best offers near extremity at the top right of the figures) and qualitative (by the respective values associated with the coordinates) vision. In particular, by observing each one of the above-mentioned maps, the results clearly show how, by the MCDM algorithm, the broker is able to select a variable number of offers in depending of the greater or lesser selectivity that is required. This selectivity is determined by the datasets and the weights assigned to the algorithm selection criteria. The goodness of the proposed solution is also demonstrated by the algorithm's ability to respond dynamically to changes in economic offers. In particular, comparing the evaluation maps having the same index (i.e., the scenarios 1, 2 or 3 respectively distinguished by evaluation map 1, 2 and 3) it is clear that it is possible to select one or more economic offers that are "good" or "satisfying", i.e., higher than established minimum values (on the Cartesian axes in Figs. 4, 5 and 6). Moreover, by observing evaluation maps, it is clear that areas where fall offers deemed advantageous are wider, as is also shown by numerical values on the axes. In particular, the maximum values listed on the x and y axes are increasing as the number of providers which are able to submit offers.This means that the offers selected by the Multi Criteria Decision Making

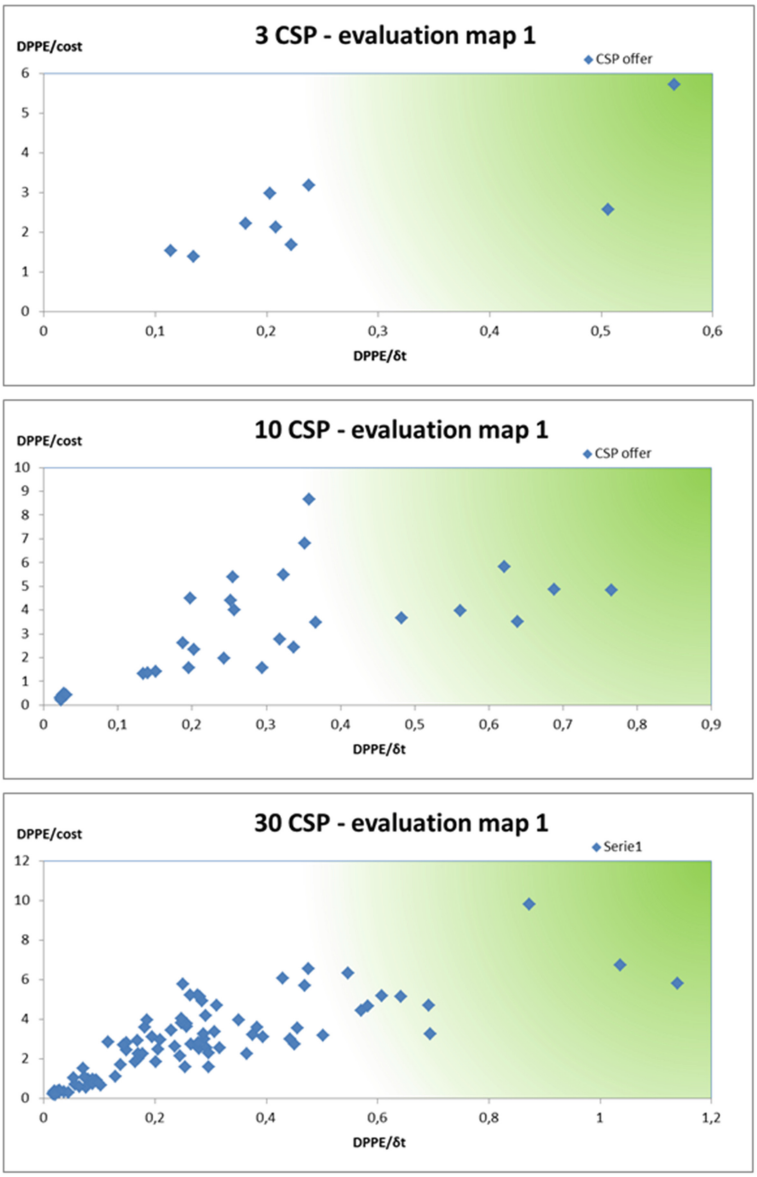


Fig. 4. DPPE/cost - DPPE/δt evaluation map for 3-10-30 CSPs. (Color figure online)

algorithm averagely provide higher values of sustainability. More specifically, graphs show higher *DPPE/cost* - *DPPE/*δt values for 30 CSP than 10 CSP or 3 CSP. The same for *DPPE/CUE* - *DPPE/*δt and *DPPE/cost* - *DPPE/CUE*.

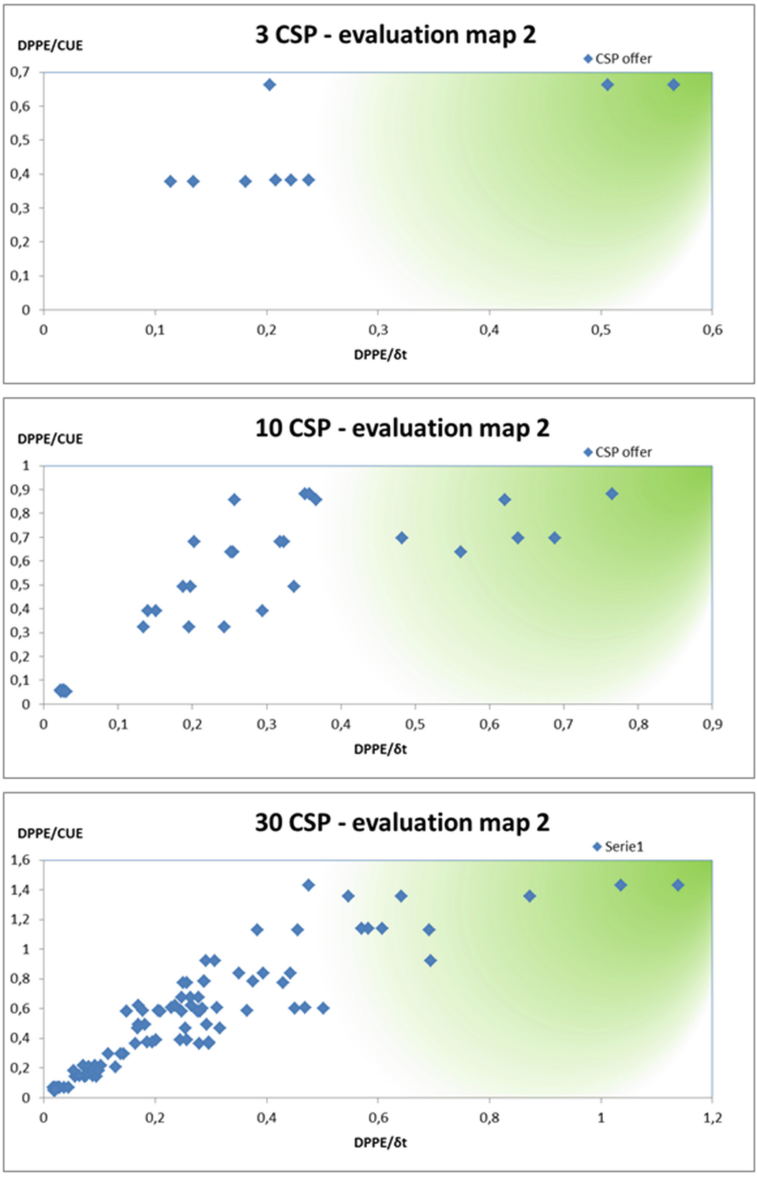


Fig. 5. DPPE/CUE - DPPE/δt evaluation map for 3-10-30 CSPs. (Color figure online)

Table 2 shown a comparison between the evaluation maps resulting from the simulations and shown in Figs. 4, 5 and 6. For each map, the first column reports the couple of thresholds on the x-y axes we chose for the derived requirements; the second one the number of CSPs; the third column the number of offers that satisfy the chosen thresholds. It is meaningful that a higher number of

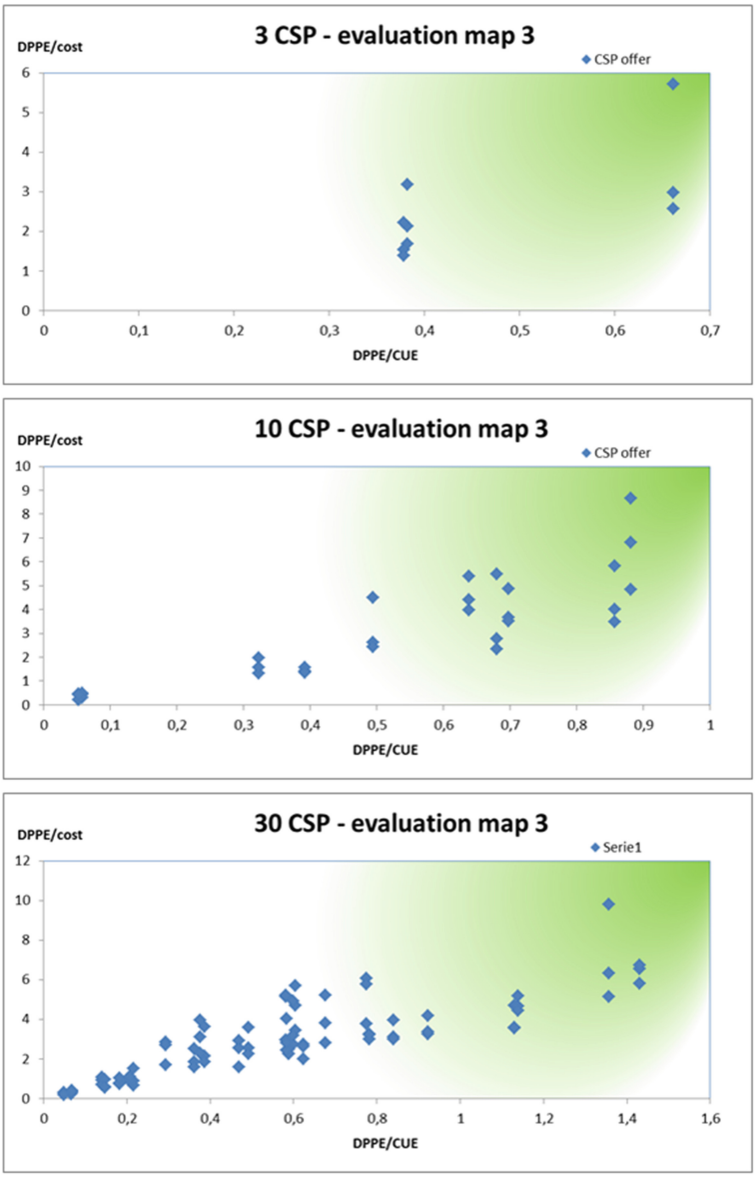


Fig. 6. DPPE/cost - DPPE/CUE evaluation map for 3-10-30 CSPs. (Color figure online)

"green" CSPs allows federation to achieve a higher level in sustainability, towards a broad federated Cloud ecosystem, which represents a good balance between sustainability, cost, and service parameters.

Table 2. Evaluation map comparison.

Evaluation map 1		
DPPE/cost - DPPE/δ**t**	CSPs Num.	Number of "green" offers
4–0,5	3	1
4–0,5	10	4
4–0,5	30	9
Evaluation map 2		
DPPE/CUE - DPPE/δ**t**	CSPs Num.	Number of "green" offers
0,6–0,5	3	2
0,6–0,5	10	5
0,6–0,5	30	7
Evaluation map 3		
DPPE/cost - DPPE/CUE	CSPs Num.	Number of "green" offers
4–0,6	3	1
4–0,6	10	10
4–0,6	30	15

7 Conclusion and Future Work

In this paper, we presented and discussed a decision system for Cloud Federation brokerage able to combine global energy sustainability and cost-saving strategies. More specifically, we presented a Multi-Criteria Decision Making (MCDM) algorithm able to discover the most convenient Cloud providers able to join a particular energy-aware Cloud Federation.

It enables multiple involved Cloud Service Providers to maximize their mutual benefit and therefore their business. Therefore, it bodes to encourage new investments in innovation thus to create a more competitive digital market. The proposed strategy can allow both small/medium-size enterprises and new companies to become Cloud service providers, both in according to specific service requirements (e.g., *Service Level Agreements (SLAs)* and sustainability metric.

By modeling, in energy-aware manner, different Cloud providers and the related economic offers, we demonstrated how the MCDM algorithm can accommodate different Cloud Federation scenarios characterized by particular energy-aware and business parameters (e.g., service price). Increasing the number of "green" Cloud service providers, a Cloud Federation can achieve highest level in sustainability, towards a broad federated Cloud ecosystem able to compete with larger providers, through a good balance between sustainability, cost, and service parameters.

However, in a big data management context, relational schemes are hard to change incrementally, and especially without impacting performance.

Therefore, in future works, we plan to investigate a strategy to reduce the environmental footprint and costs in ICT systems where analyzing unstructured, semi-structured, and polymorphic data is a priority.

Acknowledgment. This work was partially supported by EU H2020 BEACON Project G.A. 644048, 2015–2018, and by SIGMA Project - Italian National Operative Program (PON), 2007–2013. Authors would like to thank Eng. Giulio De Meo for his technical support.

References

1. Bureau, P.R.: 2015 world population data (2015). http://www.prb.org/wpds/2015/
2. European Commission: Effort sharing decision. http://ec.europa.eu/clima/policies/effort/documentation_en.htm
3. European Commission: Digital single market. https://ec.europa.eu/digital-agenda/en/digital-single-market/
4. Giacobbe, M., Celesti, A., Fazio, M., Villari, M., Puliafito, A.: An approach to reduce energy costs through virtual machine migrations in cloud federation. In: IEEE Symposium on Computers and Communication (ISCC), pp. 782–787, July 2015
5. Chandrasekar, S.: A review of literature on cloud brokerage services. Int. J. Comput. Sci. Bus. Inf. **10**(1) (2014)
6. Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., Tordsson, J., Ragusa, C., Villari, M., Clayman, S., Levy, E., Maraschini, A., Massonet, P., Muñoz, H., Tofetti, G.: Reservoir - when one cloud is not enough. Computer **44**(3), 44–51 (2011)
7. Hamze, M., Mbarek, N., Togni, O.: Broker and federation based cloud networking architecture for IaaS and NaaS QoS guarantee. In: 13th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 705–710, January 2016
8. Usha, M., Akilandeswari, J., Fiaz, A.: An efficient QoS framework for cloud brokerage services. In: International Symposium on Cloud and Services Computing (ISCOS), pp. 76–79, December 2012
9. Tang, C., Liu, J.: Selecting a trusted cloud service provider for your saas program. Comput. Secur. **50**, 60–73 (2015)
10. The Eco2Clouds Project. http://eco2clouds.eu/
11. Wajid, U., Marín, C.A., Karageorgos, A.: Optimizing energy efficiency in the cloud using service composition and runtime adaptation techniques. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 115–120, October 2013
12. Giacobbe, M., Celesti, A., Fazio, M., Villari, M., Puliafito, A.: Towards energy management in cloud federation: a survey in the perspective of future sustainable and cost-saving strategies. Comput. Netw. **91**, 438–452 (2015)
13. Scilab: Open source software for numerical computation. http://www.scilab.org/scilab
14. Ministry of Economy, Trade and Industry, Japan (METI): Enhancing the Energy Efficiency and Use of Green Energy in Data Centers. http://home.jeita.or.jp/greenit-pc/sd/pdf/ds2.pdf

A Motivating Case Study for Coordinating Deployment of Security VNF in Federated Cloud Networks

Philippe Massonet[1], Sébastien Dupont[1], Arnaud Michot[1], Anna Levin[2], and Massimo Villari[3(✉)]

[1] CETIC Research Center, Charleroi, Belgium
philippe.massonet@cetic.be
[2] IBM Research Lab, Haifa, Israel
lanna@il.ibm.com
[3] University of Messina, Messina, Italy
mvillari@unime.it

Abstract. Federated cloud networks are formed by federating virtual network segments from different cloud platforms into a single federated network. This allows virtual machines from one virtual network segment to communicate with virtual machines running on the other virtual network segments of the federated network. Federated cloud networks can be very useful for creating application specific isolated networks between clouds. In this paper we describe current work in the BEACON project to secure the federated network with a global security policy. Virtual network functions and service function chaining are used to implement the security policy. The federated cloud network security policy is described in a service manifest. This enables automated deployment and configuration of network security functions across the different cloud federation networks. The approach is illustrated with a simple case study where communications between trusted and untrusted clouds are encrypted.

Keywords: Network federation · Network function virtualisation
Service function chaining · Security

1 Introduction

There is a growing need to connect different cloud services for various purposes such as creating hybrid clouds or distributing virtual machines across different clouds. Cloud federation mechanisms enable cloud providers to collaborate and share their resources to create a large virtual pool of resources at multiple network locations. Different Cloud federation types such as cloud bursting, cloud brokering or cloud aggregation have been proposed to provide the necessary mechanisms for sharing compute, storage and networking resources even in small devices [1]. Federated cloud networking techniques provides mechanisms to federate cloud network resources, and to define an integrated cloud management

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 34–42, 2018.
https://doi.org/10.1007/978-3-319-72125-5_3

layer to deploy applications securely and efficiently within the cloud federation (see [2,3]). One approach for customising network security is to use network virtualisation technologies such as VLANs in conjunction with Network Function Virtualisation (NFV) and Service Function Chaining (SFC). By combining NFV/SFC with network virtualisation it is possible for cloud tenants to tailor the security of each of their individual virtual networks. This way, security virtual network functions (VNF) can be adapted to the specific needs of each application. VNF provide the necessary security functionalities such as deep packet inspection, encryption or firewalls. SFC can be used to compose security VNF to meet tenant's security requirements.

In this paper, we argue that the deployment and configuration of security VNF across the different clouds of a network federation should be coordinated. In the paper we provide a motivating example for coordinating the deployment of security VNF across different clouds. The specification of the global network security policy and how to implement it with VNF and SFC is defined in a service manifest. The deployment and configuration of the security VNF across the different federated cloud network segments is the responsibility of the network federation manager.

The paper is organized as follows: Sect. 2 shows in the literature the topic we are addressing here is not covered yet. Section 3 describes the motivations for coordinating the deployment of security VNF across the different federated cloud network segments. Section 4 describes the network service manifest and a motivating example. Section 5 provides a discussion on the approach and identifies future work.

2 Related Works

The section shows how in the literature there are not works able to address this important problem [4]. There is a standardization effort on SFC. IETF SFC working group [5] developed framework for SFC operations and administration. The Open Network Foundation proposed Service function chaining (SFC) architecture [6]. The architecture includes SFC network controller responsible for setting up the service chaining; Classifier responsible for classifying traffic flows and Service function forwarder, which is responsible for forwarding packets to the service functions defined in the policy table. The proposed general architecture can be applied to any network. However, in order to implement this architecture in federated heterogeneous networks, there is a need to coordinate SFC network controllers and classifiers, so they will speak the same language and deploy and control NFVs in an efficient way.

In addition to a general architecture definition, there are more detailed SFC issues addressed in the literature. For example, [7,8] address VNF service-chain placement issues in different cloud network scenarios. However, they do not address federated network scenario with its specific issues of coordinating SFCs across heterogeneous networks empowered by different virtualization technologies. The authors of [9] propose a design of a resource orchestrator that steers

the control of SFCs in a scalable way. Their orchestrator map network functions of a requested SFC to infrastructure network and compute resources. While scalability is an important issue in federated cloud, there are additional issues that have to be addressed, such as securing heterogeneous environment with different levels of trust between them.

3 Motivations

Federated cloud networks are created by joining several pre-existing network segments that are running in different clouds of a cloud federation. The cloud federation is assumed to be heterogeneous, it may be composed of different cloud platforms such as OpenStack, OpenNebula or even public clouds such as Amazon AWS. The federated cloud network enables the virtual machines that are running on the different segments of the federated network to communicate. The main benefits of federated cloud networks are the isolation they provide, i.e. they can be application specific, and the flexibility they provide to manage and extend the federated network across different cloud platforms. The main motivation of this paper is to show how a global security policy of a federated cloud network can be defined in a service manifest and implemented with security virtual network functions (VNF) and service function chaining (SFC) in each of the federation clouds. The main benefit of this approach is that the deployment, configuration and chaining of the security VNF may be automated and verified once it is deployed across the different clouds. Performing this manually across the different network segments by the network administrator could take time and be error prone if the federated cloud network is large and/or the network topology or the security policy change often. The deployment of the security VNF and their chaining across the different cloud network segments is coordinated by a component called the Federated Cloud Network Manager. The approach is illustrated with a case study that encrypts network flows if the destination is untrusted, and does not encrypt it if the destination cloud is trusted.

4 Approach

In this section we describe the approach to specifying global network security policies in a service manifest and then implementing the security policy in the federated cloud network. First we describe an example of federated cloud network.

Figure 1 shows a federated cloud network distributed across a federation of three clouds. The figure shows that each cloud has its physical compute, storage and networking resources. Network resources of the three clouds are connected over the internet. The figure shows that in each of the three clouds, and over the physical resources, there is a virtual network segment on which several virtual machines are running. The figure shows that the three virtual networks have been federated into a federated cloud network. This allows virtual machines running in the different network segments to communicate with each other.

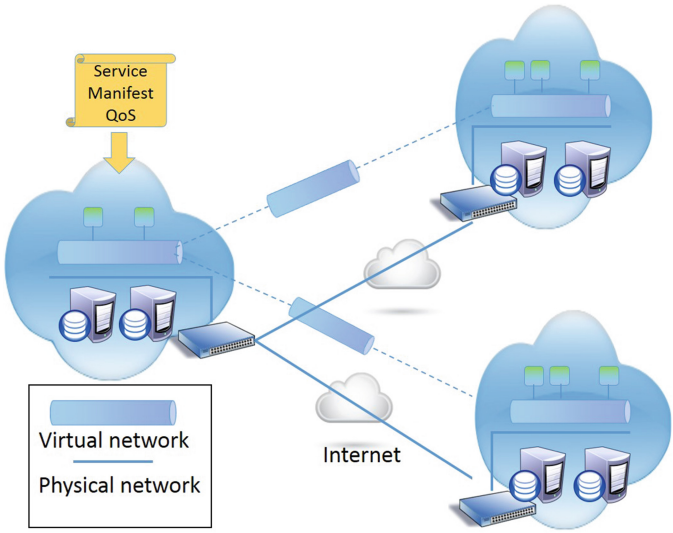


Fig. 1. A federated cloud network federation

Figure 2 shows the general structure of the federated cloud network service manifest. The service manifest contains a section describing the networking resources of the cloud federation. Part of the service manifest is dedicated to the federated cloud network global security policy. In this part the global security policy is defined by specifying for example authorized or forbidden protocols, trusted and untrusted networks, etc.

Figure 3 shows the case study with three clouds. The first two clouds are trusted and can communicate without encryption. The third cloud is a public cloud and is untrusted: communications between the first two clouds and the third cloud must be encrypted. The global security policy may be specified as follows and can later be translated into a network program:

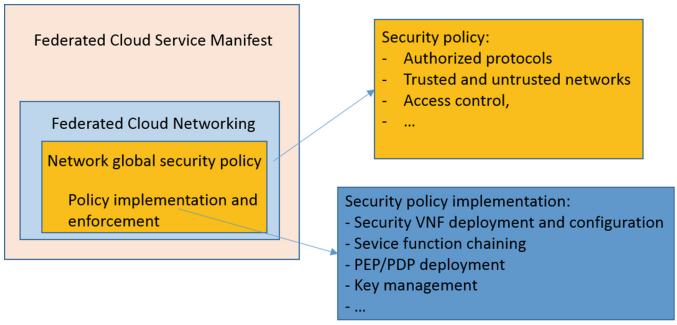


Fig. 2. A federated cloud network federation

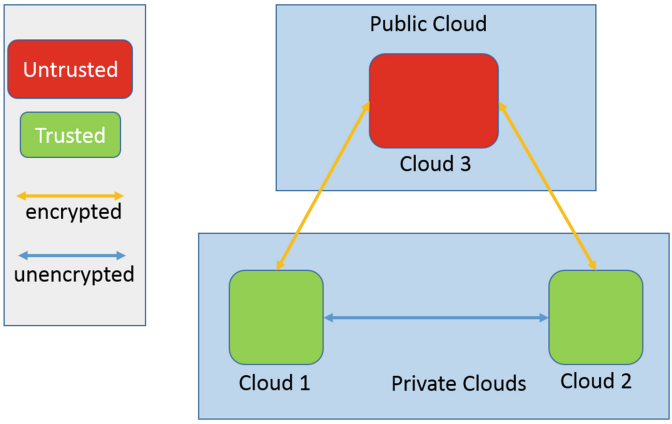


Fig. 3. Cloud case study: trusted and untrusted clouds

if destination cloud X is untrusted **then**
  encrypt network flow with public key of X
else if destination cloud Y is untrusted **then**
  encrypt network flow with public key of X
else if ... **then**
  ...
else if destination cloud Z is trusted **then**
  do not encrypt network flow
else if source cloud is untrusted **then**
  decrypt the network flow using the private key
else
  destination or source cloud is not part of the federation
end if

This policy covers both the encryption case, which must be done with the public key of the destination cloud, and the decryption case that must be done with the private key of the destination cloud.

This global network policy may be implemented with various mechanisms available in the different cloud platforms of the federation such as:

- VNF: the VNF managers of different cloud platforms provide catalogues of VNF that can be used to secure the network federation, e.g. in this case encryption.
- Traffic direction: VNF can be applied to outgoing or incoming traffic, e.g. encryption is applied to outgoing traffic and decryption is applied to incoming traffic.
- SFC: VNF need to be placed in the right sequence inside the service chain, e.g. encryption should be the last VNF in the outgoing traffic service chain and decryption should be the first VNF in the incoming traffic service chain.

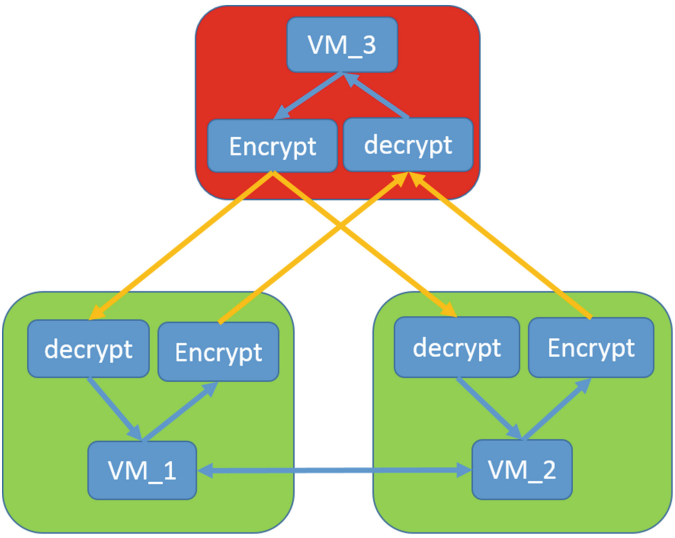


Fig. 4. SFC deployment

Figure 4 shows the resulting implementation of the above global security policy where communications between the three clouds of the federation have been secured according to the federated network security policy. Communications between cloud 1 and cloud 2 are not encrypted since both clouds are trusted. On the other hand, communications with public cloud 3 need to be encrypted because cloud 3 is untrusted. The figure shows how the encryption and decryption Virtual Network Functions have been deployed to secure communications. For example, when VM1 communicates with VM3 all network traffic is encrypted in cloud 1 and systematically decrypted by public cloud 3. All outgoing traffic from VM3 to VM1 is encrypted by cloud 3 and decrypted by cloud 1. In this case the service chaining on each cloud site amounts to an "if ... then ... else": if the traffic destination is an untrusted cloud, then the traffic has to be encrypted.

Figure 5 shows a fragment of a YAML based service manifest. It describes the encryption and decryption VNF as well as the security groups. It states that the encryption VNF should be placed at the end of the outgoing service chain and specifies which key to use based on the destination address. Conversely it specifies that the decryption VNF should be placed at the beginning of the incoming service chain and specifies the private key to be used. Security groups are sets of IP filter rules that are applied to a VM instance's networking. The traditional use of security groups is to ensure security between tenants in the cloud. However, it is also possible to use this mechanism to secure traffic in the tenant's network between clouds. Security groups applied to VM's ports by means of OpenFlow filtering rules can be used to protect VM from unauthorised access from untrusted clouds. The default security group denies all incoming traffic.

```
Cloud:
    - ID: xxx
    Name: Cloud1
    Type: Private
    - ID: xxx
    Name: Cloud2
    Type: Private
    - ID: xxx
    Name: Cloud3
    Type: Public

Security groups:
    - ID: xxx
    Name: Application 1
    ...

VNF:
    -ID: xxxxx
    Condition: DestCloudType==Public
    Type: Encrypt
    Key: xxxx
    - ID: xxxxx
    Condition: SrcCloudType==Public
    Type: Decrypt
    Key: xxxx
```

Fig. 5. YAML service manifest fragment for VNF and security groups

5 Discussion

The federated cloud networking security approach presented in this article lets tenants define, configure and chain the security VNF of their federated virtual networks. The approach enforces a global security policy on the federated cloud network. In this paper we have illustrated the need to coordinate the deployment and configuration of security VNF across the different network segments of the cloud federation. Only in this manner can the global network security policy be enforced across the federated network. This coordinated deployment of VNF is described in a service manifest and is used to automate the deployment of VNF in the different federation clouds.

For the sake of simplicity in this paper we have assumed that there were no pre-existing local network security policies that could conflict with the global security policy. In reality there will be local security policies that could conflict with the global security policy. To resolve any potential conflicts the network administrator will have to analyse pre-existing local security policies and design of the global security policy accordingly or modify local security policies

if possible. Again for the sake of simplicity we have assumed very simple service chaining in each cloud of the federation for this paper. We have assumed that no pre-existing local service chains existed. If local pre-existing service chains exist then the addition of any new security VNF should be made in a compatible manner. Verification that the modified service chain performs as expected needs to be made. Any conflicts between VNF should be resolved by the network administrator.

The global security policy of the case study was presented in Sect. 4 in the form of a network program can be reused. For instance when new network segments are added to the federation the network program can be applied to deploy, configure and chain the security VNF. In this manner the same global security policy will be enforced on all network segments of the federation. The approach was illustrated here on a specific case study but future work will look other case studies to identify other network programming abstractions.

6 Conclusion and Future Work

Federating cloud virtual networks across different clouds provides much needed mechanisms for administrators, tenants and applications to create customized and isolated networks. Federated cloud networks extend existing cloud federation mechanisms such as cloud bursting, cloud brokering or cloud aggregation. This paper described a high level approach for customising federated cloud network security with Network Function Virtualisation (NFV) and Service Function Chaining (SFC). It was argued that the deployment and configuration of security VNF and their chaining across the different federated cloud networks should be coordinated. The approach was illustrated with a motivating case study showing that security VNF such as encryption and decryption may conditionally be applied based on destination and source IP addresses. Future work involves developing a network federation manager and experimenting with service chaining in OpenStack.

Acknowledgment. This work has been supported by the BEACON project, grant agreement number 644048, funded by the European Union's Horizon 2020 Programme under topic ICT-07-2014.

References

1. Celesti, A., Fazio, M., Giacobbe, M., Puliafito, A., Villari, M.: Characterizing IoT cloud federation, Le Régent Congress Centre, Crans-Montana, Switzerland. IEEE Computer Society (2016)
2. Moreno-Vozmediano, R., et al.: BEACON: a cloud network federation framework. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 325–337. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33313-7_25
3. Massonet, P., Levin, A., Celesti, A., Villari, M.: Security requirements in a federated cloud networking architecture. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 79–88. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33313-7_6

4. Banerjee, S., Shaw, R., Sarkar, A., Debnath, N.: Towards logical level design of big data. In: IEEE 13th International Conference on Industrial Informatics (INDIN), pp. 1665–1671, July 2015
5. Aldrin, S., Krishnan, R., Pignataro, N.A.C., Ghanwani, A.: Service function chaining operation, administration and maintenance framework. In: IETF RFC, February 2016
6. L4-l7 service function chaining solution architecture. In: ONF TS-027. Version 1.0, June 2015
7. Gupta, A., Habib, M.F., Chowdhury, P., Tornatore, M., Mukherjee, B.: On service chaining using virtual network functions in network-enabled cloud systems. In: 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pp. 1–3, December 2015
8. Mehraghdam, S., Karl, H.: Specification of complex structures in distributed service function chaining using a YANG data model. CoRR, abs/1503.02442 (2015)
9. Sahhaf, S., Tavernier, W., Czentye, J., Sonkoly, B., Sköldström, P., Jocha, D., Garay, J.: Scalable architecture for service function chain orchestration. In: 2015 Fourth European Workshop on Software Defined Networks, pp. 19–24, September 2015

The Big Bucket: An IoT Cloud Solution for Smart Waste Management in Smart Cities

Maurizio Giacobbe[1(✉)], Carlo Puliafito[2(✉)], and Marco Scarpa[1(✉)]

[1] Department of Engineering, University of Messina,
Contrada Di Dio, 98166 Messina, Italy
{mgiacobbe,mscarpa}@unime.it
[2] DIEEI, University of Catania,
Viale Andrea Doria 6, 95100 Catania, Italy
carlopulia@gmail.com

Abstract. Research and industries are devoting a great effort in getting cities and communities smarter, thus to improve citizens' *Quality of Life (QoL)* and paying serious attention to e-government and e-inclusion processes. This is a strategic but also very complex objective that involves both governance and citizens to address many challenges. Following this line, this paper discusses the necessity for new smart waste management systems and presents a comprehensive state of the art on the use of the **Internet of Things (IoT)** for smart waste recycling. In particular, we present and argue the Big Bucket IoT Cloud environment, where smart dumpsters are equipped with low-cost sensors and open source easy-to-use hardware and software. Its architectural model is discussed and compared with other existing solutions in the future perspective.

Keywords: Smart cities · Smart communities · Quality of Life
Smart waste management · Recycling systems · E-government
E-inclusion · IoT · Cloud Computing

1 Introduction

A smart city is an urban multi-disciplinary development vision where *Information and Communication Technology (ICT)* integrates both existing and new city's assets mainly in order to improve citizens' **Quality of Life (QoL)**. City's assets mainly include, but not limited to, governmental departments, public and private transportation systems, medical-hospital centers, power plants, educational centers (e.g., schools, universities, colleges, museums, libraries), water supply network, waste management and other community services. ICT allows city officials to directly interact with the community, for example by applying new e-government rules and laws and by involving people in new e-inclusive policies and processes. Moreover, ICT allows city officials to collect information concerning the city infrastructure in order to monitor (in real-time or on-demand) what is happening in the city and specifically in which area, where and how the

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 43–58, 2018.
https://doi.org/10.1007/978-3-319-72125-5_4

city is evolving, and how to implement improvements to assure a better citizens' QoL.

Among the ICTs, the *Internet of Things* is a "hot" topic. The *ITU-T SG20 Study Group on IoT and its applications including smart cities and communities (SC&C)* [1] motivates IoT as having the potential to change people's lifestyle and the way they interact with the surroundings, especially in smart cities and communities (SC&C). In this regard, it is expected that IoT will have a significant impact on the above-mentioned key infrastructural elements pertaining to cities, including the QoL, and environment as well as on society and the economy in general.

This paper specifically focuses on the use of IoT for **smart waste management** and **recycling**. Over-consumption and waste it creates is a growing problem for today's society. Therefore, monitoring the *lifecycle* of waste is very important for almost two main reasons: to reduce epidemiological risks and to improve QoL. About these objectives, a city can be defined "smart" only if it includes a really efficient waste management assets. City officials can use technological solutions based on the IoT and Cloud paradigm to gather data from smart objects dislocated in several areas of the city, to analyze them and to choose the best solutions to improve existing services and to implement new and better QoL-oriented services. For example, by collecting data from monitored zones it is possible to determine if a specific area is well served or not, what are the most important critical situations (e.g., if dumpsters are sufficient in number or not), and how to prevent epidemiological problems.

The *European Waste Catalogue (EWC)* is a hierarchical list of waste descriptions established by $CommissionDecision2000/532/EC$. Commission Decision of 18 December 2014 amending Decision 2000/532/EC on the list of waste pursuant to Directive 2008/98/EC of the European Parliament and of the Council. This results in a new EWC document as reported in [2]. It is divided into twenty main chapters each of which has a two-digit code between 01 and 20. Individual wastes within each chapter are assigned a six figure code. In particular, the *household* wastes are generally coded under Chapter 20 as shown in Table 1, even if there are some wastes where it is more appropriate to use another code.

Therefore, industries and governmental entities, along with scientific research and academia, are interested to develop innovative technological solutions in order to allow an efficient **waste management in smart cities**, by classifying waste using the EWC. Generally, they take into account three key factors when thinking about how to recycle: the **3 R's**: **Reduce**, **Reuse**, **Recycle**.

To this purpose, in this paper we propose an integrated IoT Cloud framework where a new smart dumpster, named **Big Bucket**, is integrated into a technological platform designed to provide waste services safer for both the environment and humans. More specifically, the proposed solution is addressed to the household waste collection (i.e., the above-mentioned wastes within Chapter 20 in the EWC).

An exemplified scenario is shown in Fig. 1, where several Big Buckets are geo-localized in different zones and equipped with sensors and connectivity to

Table 1. EWC Chapter 20 - Municipal wastes (household and similar commercial, industrial and institutional wastes) including separately collected fraction.

EWC Chapter 20 table	
Code	Waste description
20 01	**Separately collected fractions (except 15 01)**
20 01 01	Paper and cardboard
20 01 02	Glass
20 01 08	Biodegradable kitchen and canteen waste
20 01 10	Clothes
20 01 11	Textiles
20 01 13[a]	Solvents
20 01 14[a]	Acids
20 01 15[a]	Alkalines
20 01 17[a]	Photochemicals
20 01 19[a]	Pesticides
20 01 21[a]	Fluorescent tubes and other mercury-containing waste
20 01 23[a]	Discarded equipment containing chlorofluorocarbons
20 01 25	Edible oil and fat
20 01 26[a]	Oil and fat other than those mentioned in 20 01 25
20 01 27[a]	Paint, inks, adhesives and resins containing hazardous substances
20 01 28	Paint, inks, adhesives and resins other than those mentioned in 20 01 27
20 01 29[a]	Detergents containing hazardous substances
20 01 30	Detergents other than those mentioned in 20 01 29
20 01 31[a]	Cytotoxic and cytostatic medicines
20 01 32	Medicines other than those mentioned in 20 01 31
20 01 33[a]	Batteries and accumulators included in 16 06 01, 16 06 02 or 16 06 03 and unsorted batteries and accumulators containing these batteries
20 01 34	Batteries and accumulators other than those mentioned in 20 01 33
20 01 35[a]	Discarded electrical and electronic equipment other than those mentioned in 20 01 21 and 20 01 23 containing hazardous components
20 01 36	Discarded electrical and electronic equipment other than those mentioned in 20 01 21, 20 01 23 and 20 01 35
20 01 37[a]	Wood containing hazardous substances
20 01 38	Wood other than that mentioned in 20 01 37

(*continued*)

Table 1. (*continued*)

20 01 39	Plastics
20 01 40	Metals
20 01 41	Wastes from chimney sweeping
20 01 99	Other fractions not otherwise specified
20 02	**Garden and park wastes (including cemetery waste)**
20 02 01	Biodegradable waste
20 02 02	Soil and stones
20 02 03	Other non-biodegradable wastes
20 03	**Other municipal wastes**
20 03 01	Mixed municipal waste
20 03 02	Waste from markets
20 03 03	Street-cleaning residues
20 03 04	Septic tank sludge
20 03 06	Waste from sewage cleaning
20 03 07	Bulky waste
20 03 99	Municipal wastes not otherwise specified

[a]Hazardous (special) wastes.

communicate with a IoT Cloud framework. This one is able to analyze data and to implement algorithms to help waste operators to make the best choices for their services. For example, through remote monitoring of the full level of each dumpster and for each area of the city, it is possible to optimize service scheduling and fleet management, that is the collection schedules, management of the operators and choosing suitable vehicle for each zone. Moreover, real-time data can be useful to provide public information on air quality, or to realize *e-inclusion* contexts through collaboration and *Open API*.

The proposal is part of the **#SmartME crowd-founding project** [3], a smart city infrastructure in the city of *Messina*, where IoT solutions and services are being integrated in one coherent and single development framework. An Open Data platform has been set up through the employment of low-cost micro-controller boards, equipped with sensors and actuators and installed on buses, lamp posts, and buildings of local institutions, all over the urban area. Thanks to such infrastructure, it is possible to collect data and information to design advanced services for citizens who may take part in this network, through smart-phones and other mobile devices by which it is possible to interact with objects and may even themselves turn into data producers.

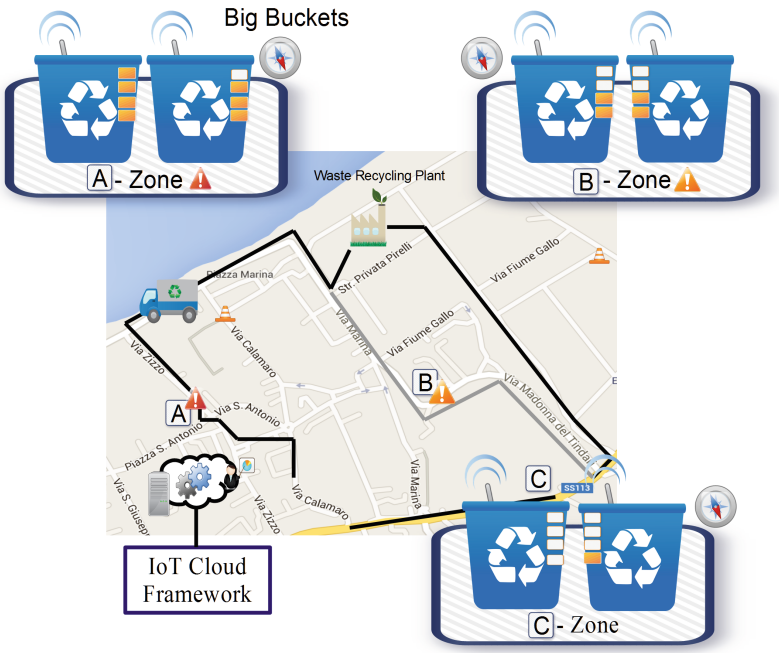


Fig. 1. Exemplified waste management scenario with the Big Buckets smart dumpsters.

2 State of the Art in Smart Waste and Recycling Systems

Nowadays, there are some interesting solutions in smart waste and recycling systems field. However, they do not currently provide support to the city officials in order to simultaneously optimize waste management and to improve QoL in specific areas.

This section presents a large overview on the *state-of-the-art* in smart waste and recycling systems solutions. The following paragraphs respectively introduce several important scientific contributions in Literature and other business solutions. In particular, both architectural models and sensors are presented and argued.

2.1 Scientific Literature

In [4] the authors motivate and propose an IoT-enabled system architecture to achieve a dynamic waste collection to deliver to recycling plants. It uses a *top-k* query based dynamic scheduling model which addresses the challenges of near real-time scheduling driven by sensor data streams. Moreover, the authors present a very interesting case of study applied to the city municipality of St. Petersburg in Russia. If compared with traditional waste collection systems, their proposed IoT-enabled system architecture is a strong improvement in that

context. However, a weakness can be represented by the use of SQL databases that are not proper for a *Business Analytics as a Service* context, where instead NoSQL databases (e.g., MongoDB) are optimal.

A model which incorporates a robust dynamic routing algorithm is presented in [5]. Specifically, the proposed model proved to be robust in case of truck malfunction due to truck overloading or damage.

An interesting study on domestic (i.e., household) waste treatment and disposal is presented in [6]. The authors configure several technologies at different architectural levels (i.e., perceptual, transport and application). In particular, perceptual layer includes Radio Frequency IDentification (RFID), both wireless and wired sensors, and Global Positioning System (GPS).

In [7] the authors present an IoT based household-waste container for future smart cities, called *Recyclable, eco-Friendly, on-Demand Bin (ReDBin)*. They discuss the related architectural model and compare it to two other approaches. Their proposed solution certainly represents a step forward in household-waste collection because it is an example of city-wide eco-system able to integrate home-based and city-wide technology.

The aim of the EU Framework 7 **BURBA** Project [8] is the development of an automatic system to be used for intelligent waste management, by using Radio Frequency IDentification (RFID) and cell-based-phone Locations Based Service (LBS) abilities. The project involves three different municipalities: $Camogli(Italy)$, $Santander(Spain)$ and $Rzeszow(Poland)$.

2.2 Business Solutions

The **Bigbelly Solar Smart Waste & Recycling System** combines the power of Cloud computing with smart solar-powered, compacting waste and recycling stations. It provides its services to more than 1500 customers in 47 countries [9]. At the edge of the Bigbelly system, there are the stations, smart garbage collectors exchanging information with Servers. This information can then be viewed via the related CLEAN Management Console on the Web or on iOS and Android devices through the relative apps. The stations, which are essentially made of recycled steel and plastics, mount a solar panel that is used to charge an internal 12 V battery. The stations keep functioning for more than 72 h without direct sunlight and electronic components support a temperature range of −40 °F to +185 °F (−40 °C to +85 °C). It is available in two version, i.e., standard and high capacity, the second one integrating a compactor which ensures a compaction ratio of up to 5-to-1. Thanks to the compactor, it is possible to store the equivalent of about 150 gal (568 L) of non compacted trash, while the first model can store 50 gal (227 L). Both the stations incorporate an infrared emitter and sensor which periodically monitors trash level and, once the light beam is broken, informs a built-in microprocessor. First of all, this one activates one of the three external LEDs, indicating trash level to people: green means that the bin is close to be empty, yellow that it is quite full, while red means that the bin is completely full. Secondly, when the compacting system is expected, the microprocessor initiates a compaction cycle: the higher is the trash level, the

more frequent will be the compaction cycles. Last but not least, when necessary, the microprocessor can notify the personnel to collect the garbage, through General Packet Radio Service (GPRS) connectivity. The Bigbelly is also able to check the machine status and incorporates a GPS receiver. Furthermore, a wireless repeater is installed on the bottom of each station thus to provide free public Wi-Fi hotspots. However, costs are a weakness, because the cost of one station is of about 4000$, to which is to be added a cost of around 500$ per unit for battery replacement and about other 20$ per year for maintenance against accidental damages and vandalism. These costs are a problem for that cities in hardship, in order to realize a homogeneous and large-scale waste collection.

Enevo is a proactive company in smart waste and recycling systems field. Its solution, called Enevo ONe [10], is somehow different from Bigbelly's since Enevo does not produce the whole smart bin, but only the smart ultrasonic wireless sensors to install inside it. Enevo provides its customers with similar services to the ones offered by Bigbelly: automation of waste collection planning and optimization of the routes, all possible thanks to a Cloud service in the background, the Enevo ONe Web interface and the sensors themselves. As an example of Enevo ONe's efficiency, the city of Nottingham, England, is rolling out the service. With a population of 727977 citizens, Nottingham had 660 daily collections. But using the Enevo sensors, it could cut that number down to 68 a day, or 89% less [11]. Enevo's sensors can be installed in over 100 container types and, by now, the supported waste types are: mixed, paper and cardboard, glass, metal, textile and Waste Electrical and Electronic Equipment (WEEE). The sensors incorporate high-performance lithium batteries which, in optimal circumstances, can last over 10 years. Actually, Enevo's sensors do not sense only trash level, but also temperature (very useful e.g. in case of fire) and movement (e.g. in case of vandalism). They support temperatures in a range of $-40\,^{\circ}\mathrm{F}$ to $+185\,^{\circ}\mathrm{F}$ ($-40\,^{\circ}\mathrm{C}$ to $+85\,^{\circ}\mathrm{C}$) and are also water and dust proof (IP66). Enevo uses a proprietary ultrasonic sonar technology for its smart sensors (smart because they run a software which, among other things, is used to determine trash level even if the surface of the contents is unevenly shaped). Measurements are performed at configurable intervals, usually once per hour. Substantially, Enevo produces two sensors, the WE-008 and the WEL-001 sensors. While the first one, which has an accuracy of ± 5 cm (± 1.96"), can deal with both solid and liquid contents, the second, which has an accuracy of ± 1 cm (± 0.39"), is specialized for liquids. Measurements are communicated to the Servers using GSM. Customers can check the status of their containers at any time through the Enevo ONe Web interface. Alarms can be set, via the web based system, to inform the user of detected events and wireless configuration and remote software updates are possible. Enevo has recently joined **LoRa™ Alliance**, an open, non-profit association of members collaborating together and sharing experience to drive the success of the LoRa protocol, LoRaWAN™, as the open global standard for secure, carrier-grade IoT **Load-Power Wide-Area (LPWA)** [12] connectivity. Enevo's solution price is calculated per container and is paid monthly. Furthermore, sensors are fully maintenance free.

Apart from the above discussed infrared and ultrasonic level sensors, nowadays other level measurement techniques do exist. However, the use of more complex and expensive sensors is justified only for *special waste* in industrial environments, where higher risks need more restrictive (i.e., precise) controls, furthermore in real-time and following a precautionary approach. The *capacitive level sensing* is one of these techniques. If the material to measure the level is non conductive, this technique exploits the fact that the introduction of a dielectric different from air between the electrodes of a capacitor causes a change in capacitance. Thus, capacitance is indicative of the material level. Whereas, if the material is conductive, it will act as one of the two capacitor electrodes and, as the material level increases, the area of the capacitor plates does the same, varying the capacitance which is thus still indicative of the waste level. In the first scenario, the sensing probe is one of the two conductors of the capacitor, while the other one is the wall of the metallic tank or a conductive reference inserted into the tank if the latter is non conductive. In the second scenario, the dielectric is represented by an insulating sheath covering the conductive probe, while the waste material is the other conductor of the capacitor. **Omega Engineering** produces the LV3000/LV4000 series [13] level measurement probes and, of each series, it offers several different models. In general, the main common features are the accuracy of 0.5%, the operating temperatures in a range of 14 °F to 248 °F (−10 °C to 120 °C) and the maximum probe length of 6 ft (1.8 m). The starting price of the products of these two series is 846$. Another type of level sensors is the hydrostatic one. This kind of sensors cannot be applied to solids, as they are used to measure the pressure created by a liquid. The greater the level of the liquid, the greater the pressure. **First Sensor**'s CTE/CTU/CTW9000...CS series [14] is an example of submersible hydrostatic liquid level sensors which allow for high compatibility with many industrial liquids, even the corrosive ones. These sensors work in pressure ranges of 0–100 mbar up to 0–5 bar, depending on the model. The operating temperature range is of 14 °F to 158 °F (−10 °C to 70 °C). The sensor length is of 0.42 ft (12,9 cm) without considering the cable whose length depends on the model.

Sotkon waste systems is present in 12 countries in the world. It provides several waste management systems, mainly distinguished in underground and semi-underground, called **Sotkis** intelligent systems [15]. Its Sotkis Access solution is a *Pay-As-You-Throw (PAYT)* system able to collect data and to manage information about who deposited what and what kind of waste, when and which quantity. An access key is used by any of the different users. The data transference can be made by a complete GPRS/GSM system, and an Internet portal is managed by the municipality. Sotkon provides smart dumpsters which can also provide municipalities and citizens with many other services which improve life in the cities.

Envac *vacuum technology* is presented [16] as a sustainable, flexible and cost efficient way to handle waste generated in almost any urban environment (e.g., household waste, street litter, kitchen waste, soiled linen and hospital waste).

Libelium is a spanish company established in 2006 which designs and manufactures hardware and a complete software development kit (SDK) for wireless sensor networks.

It provides an open wireless sensor platform, called *Waspmote* [17], and considers different modules to be integrated in order to implement the main communication protocols: for example, the XBee 802.15.4 OEM Digi module implements the IEEE 802.15.4 standard, while the XBee ZB Digi module allows for Zigbee connectivity. Furthermore, other modules exist for the LoRaWAN, Sigfox, Wi-Fi, GPRS/3G and Bluetooth v4.0 connectivities. Data gathered by the Waspmote nodes are sent to the Cloud by Meshlium, the gateway router designed to connect Waspmote sensor networks to the Internet via Ethernet, Wi-Fi and 3G interfaces.

The above board is designed to incorporate specific sensors to be applied in a certain context. A possible sensor board to be employed in a smart dumpster is the *Smart Cities Sensor Board* which can integrate temperature, humidity, luminosity, dust, noise and level sensors, to monitor the relative parameters outside or inside the container. Another board is the *Gases PRO Sensor Board* which can monitor internal or external gases such as Carbon Monoxide (CO), Carbon Dioxide (CO2), Methane (CH4), Ammonia (NH3) and others. Pollution monitoring and display to the citizens is essential to compare the impact of measures taken by municipalities and public institutions and raise public awareness. For example, monitoring of pollution in *Stockholm* city center made its citizens approve in a referendum a congestion tax for accessing to downtown. The results were a 22% reduction in CO2 emissions and a 18% reduction in the average time of jams.

The goal of the European Project Life named **Identification DEterminatioN Traceability Integrated System for Waste Electrical and Electronic Equipment (IDENTIS WEEE)** [18] was to double the collection of WEEE (e.g. smart-phones, light bulbs, electronic toys, TVs, appliances in general) which can cause potential environmental problems if not properly disposed off, and contain valuable materials (e.g., iron, aluminum, glass, tungsten, palladium, etc.) that can be recovered and reused. To achieve this aim, fundamental features of the system are: (i) the complete traceability of electrical and electronic equipment from the moment of delivery to that of recovery; (ii) the identification of each user and the items delivered, to prevent the illegal trafficking of e-waste (valuable and potentially dangerous for the raw materials they contain); (iii) the monitoring of filling for an optimized management of the collection service. It started at the beginning of 2013 in some parts of Emilia-Romagna (Italy) and in Spain and ended up in 2015.

3 The Big Bucket Solution

This section shows the main objectives the proposed Big Bucket solution aims to realize. Big Bucket is proposed as a system of selective collection of waste based on modern technologies of the IoT, thus to mainly address the problem

of inefficiency in traditional waste collection systems. More specifically, once data are collected, the Big Bucket solution provides services for city officials in order to:

- optimize collection of waste and therefore contributing to cost-saving and users' satisfaction issues, by intervening on the basis of the status (e.g., full/empty, content waste) of the Big Bucket smart dumpsters which are distributed throughout a specific monitored area;
- optimize fleet management for waste collection vehicles, that is the displacement of wheeled means of transport, thus consequently reducing traffic and pollution (i.e., improving air quality), and choosing suitable vehicles to serve a specific area;
- carry out a monitoring of key environmental parameters (temperature, noise, humidity, air quality) to improve the QoL in urban contexts;
- monitor the type of produced waste (Identity card of the waste) to perform statistical analysis on citizens' QoL, thus to identify critical condition from the epidemiological point of view;
- achieve e-inclusion contexts through collaboration and Open API (see Fig. 2), thus reducing *ecological footprint*. The Big Bucket solution can also operate as an enabler for people to actively participate in the economy and society of tomorrow.

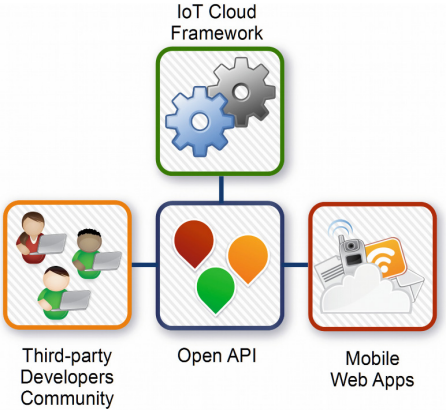


Fig. 2. Open API chart.

3.1 The Big Bucket Smart Dumpster

The proposed solution integrates the Big Bucket smart dumpsters as smart IoT units. Each one of them mainly consists, but not limited to, in a self-powered instrumented dumpster for waste collection.

The Big Bucket smart dumpster differs from the others in marketplace mainly because it is designed to be equipped with open-source and low-cost technologies

based on easy-to-use hardware and software. Moreover, as previously reported in the *state-of-the-art* section, the solution which are present in scientific literature or commercially available are almost always unable to simultaneously optimize waste management and to improve QoL in specific areas.

Actually, the Big Bucket is thought to serve two different typologies of separate waste collection: **community collection** (for public open spaces instead of traditional old dumpster) and **curbside collection** (for household waste), each one with the respective version of the Big Bucket, both essentially made of recycled steel and plastics.

Moreover, the actual designed version of Big Bucket smart dumpster is based on *Arduino* [19], an open source prototyping platform for both industrial and academia uses. However, just thanks to the use of low-cost sensors with both open source easy-to-use hardware and software solutions, the on-board instrumentation can be easily updated with news produced by Arduino but not limited to this provider.

Table 2 shows the main technical characteristics of both the above-introduced versions. In particular, chemical and air quality sensors allow the Big Bucket smart dumpster to be an *ambient control unit* able to monitor the type of produced waste and to identify possible epidemiological risks. To reduce the possibility of theft, a blocking system is provided for all the Big Bucket versions.

3.2 Integration of the Big Bucket Smart Dumpster in the Stack4Things Framework

In the last years, the IoT has become one of the most attractive fields in ICT thanks to the wide diffusion of smart and heterogeneous sensor-and actuator-hosting platforms. Adequate technologies are needed to afford challenges such as management, organization and coordination of these devices in order to take advantage of their features and to facilitate the implementation of higher-level services based on them. In this direction, the integration of IoT and Cloud Computing is one of the most effective solutions. More specifically, the proposed Big Bucket Smart Dumpster is provided with a *"smart" IoT board* and connectivity to be integrated in a Cloud IoT *Sensing-and-Actuation-as-a-Service (SAaaS)* framework in order to realize a smart waste management module.

The *Mobile and Distributed Systems Lab (MDSLab)* at the *University of Messina, Italy,* developed **Stack4things**, an *OpenStack* extension which manages sensing and actuation resources, remotely controlling nodes as well as virtualizing their functions and creating network overlays among them. Thus, the aim is not that of using Cloud Computing simply to manage data, but to shift to a *SAaaS* perspective in which to provide developers or final users with actual or virtualized sensing and actuation resources. More specifically, Stack4Things is an OpenStack-based and Cloud-oriented horizontal solution providing IoT object virtualization, customization, and orchestration [20].

Table 2. The Big Bucket smart dumpster. Characteristics.

Version	Community	Curbside
Power	Solar Photovoltaic (PV) Panel and PV Management Unit (with energy storage unit).	
Capacity	**Standard:** 50 gal / 190 l (designed for lower-capacity, e.g. compost); **High:** 150 gal / 570 l (designed for higher-capacity).	**Medium:** < 63.4 gal / 240 l, **Small:** <31.7 gal / 120 l
Portability	blocking system	blocking system + 2 wheels
Connectivity	public free wifi + NFC + bluetooth	protected wifi + NFC + bluetooth
IT Board	Arduino based	
NFC	Near Field Communication (external). For citizens, and for tourists (with registration). In Alternative Smart Card Technology.	
Smart Card (optional)	External. For citizens and Special Smart Card for tourists (ticket service);	
Sensors		
Level	Internal IP66 Full/Intermediate/Empty level sensor. Ultrasonic sensors (solution 1), Laser sensors (solution 2). External strip LED.	
Chemical	Internal emission level sensors of Smoke, Ammonia, Methane, but not only.	
Air Quality	External of Temperature, humidity, noise, air quality, carbon dioxide (CO2), but not only.	

The Technologies Involved. Open source technologies and standards are involved in the implementation of Stack4Things. As said above, the core is OpenStack, which is an open source framework for creating and managing private and public Clouds, since it controls compute, storage and networking resources. At the edge of the system, the IoT nodes are **Arduino YUN-like boards**. These are open source platforms that can interact with the ambient thanks to digital and analog I/O pins and can connect to the Internet through the Ethernet and Wi-Fi interfaces. Low-power microcontroller (MCU) and microprocessor (MPU) units are integrated, with the latter running an *OpenWRT* [21] Linux distribution and able to directly access the MCU I/O pins.

Moreover, the *shield* is an element that can be plugged onto a board to give it extra features. The Arduino GSM shield is able to connect the Arduino board to the Internet using the GPRS wireless network.

A Tx/Rx *Bluetooth HC-05 Class 2* module allows to cover a *Personal Area Network (PAN)* with a 2.5 mW (4 dBm) maximum power and a 10 m Operating range. Compared to Bluetooth Classes 1 and 3, our choice is the best compromise between energy-saving and operating range.

Another technology used in Stack4Things is *WebSocket*, which provides a bidirectional, full-duplex, persistent connection from a Web browser to a Server, in which the Client, but also the Server can send content at any given time to the other. So, the Server does not need anymore to be called by the Client in order to send it messages. Communications are done over TCP port number 80, which is of benefit for those environments which block non-Web Internet connections using a firewall. The last technology involved in Stack4Things is *Web Application Messaging Protocol (WAMP)*, an open standard WebSocket sub-protocol that provides two application messaging patterns, Remote Procedure Calls (RPC) and Publish/Subscribe, in one unified protocol. In WAMP, the *Unified Application Routing* is responsible of brokering Publish/Subscribe messages and routing remote calls. Figure 3 shows the Stack4Things overall architecture. It focuses on the communication between end users, on one side, and an Arduino YUN-like board on the other. On the board side, the *Stack4Things lightning-rod* runs on the MPU and interacts with the OS tools and services, and with sensing and actuation resources through I/O pins. It represents the board's point of contact with the Cloud infrastructure, allowing the end users to manage the board resources even if they are behind a NAT or a strict firewall. This is ensured by a WAMP and WebSocket-based communication between the Stack4Things lightning-rod

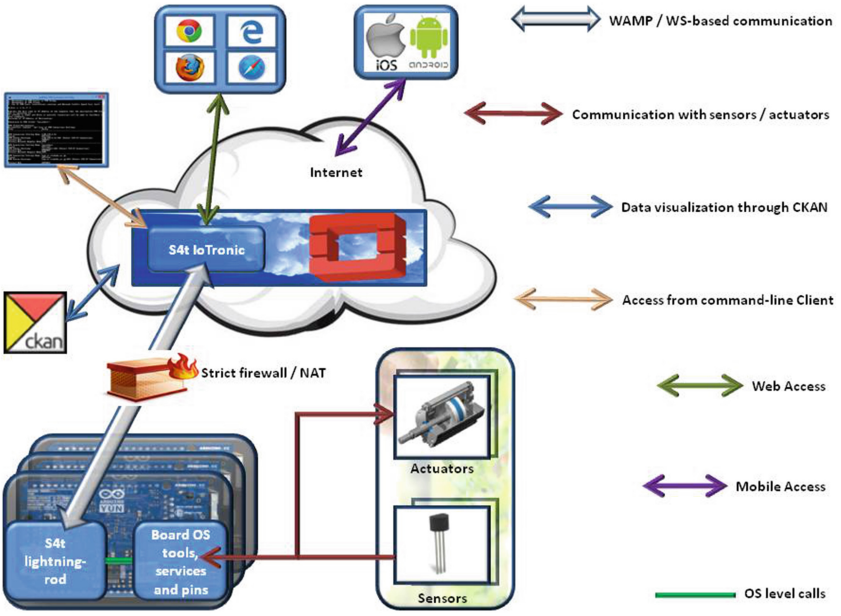


Fig. 3. The Stack4Things overall architecture

and its Cloud counterpart, the *Stack4Things IoTronic service* [22], which is implemented as an OpenStack service. The end users can interact with the Stack4Things IoTronic service (and thus with the Stack4Things lightning-rod) both via a command line based Client, the *Stack4Things command-line* Client, and a Web browser through a set of REST APIs provided by the Stack4Things IoTronic service. This one comprises a set of agents: among others, the *IoTronic registration agent* deals with nodes registration to the Cloud while the *IoTronic command agent* deals with nodes management and command delivering.

For each monitored zone, each Big Bucket dumpster sends data in real-time to the *collector node* for that zone which, in turn, remotely sends data gathered

Table 3. A comparison between the proposed Big Bucket solution, ReDbin, BURBA, Bigbelly and Sotkis.

Comparative table					
Comparative fields	ReDbin	BURBA	Bigbelly	Sotkis	Big Bucket
Self-powered by solar panel	Y		Y		Y
Ground-level collection	Y	Y	Y		Y
Underground collection				Y	
Dumpsters geolocation	Y	Y	Y	Y	Y
Level monitoring	Y	Y	Y	Y	Y
Waste type monitoring				Y	Y
Air quality monitoring	Y				Y
Real-time data	Y		Y	Y	Y
GSM/GPRS	Y		Y	Y	Y
WIFI	Y		Y		Y
RFID		Y		Y	
NFC					Y
Bluetooth					Y
Remote control	Y		Y	Y	Y
Cloud based	Y		Y	Y	Y
Open hardware/software					Y
Mobile apps	Y	Y	Y	Y	Y
Collaboration and open API			Y		Y
Web access	Y	Y	Y	Y	Y
Reporting	Y	Y	Y	Y	Y
Routes optimization	Y		Y	Y	Y
Self-alarming	Y				Y

from the connected Big Buckets to the Stack4Things [23] for the next *business analysis* phase. Data collected by the nodes are open and users can visualize them through *Comprehensive Knowledge Archive Network (CKAN)*, an open source data portal software. CKAN is built with *Python* on the back-end and *Javascript* on the front-end and its database engine is *PostgreSQL*. It also offers a powerful API that allows third-party applications and services to be built around it.

Comparison of the Big Bucket with Existing Household Waste Management Solutions. A comparison between the proposed Big Bucket solution, ReDbin, BURBA, Bigbelly and Sotkis is shown in Table 3, where the above-mentioned household waste management solutions are compared on the basis of several functionalities and services.

4 Conclusion and Future Work

In past years, waste collection was treated in a static way, also limiting the management only to the collection and without paying close attention to the citizens' *Quality of Life (QoL)*. Nowadays, the proliferation of sensors and actuators forming "smart" IoT contexts in smart cities enables new dynamic approaches, furthermore improving both waste management and QoL.

This paper addresses the challenge of specifying an innovative solution in smart waste and recycling systems. The proposed Big Bucket solution integrates "smart" dumpsters offering Wi-Fi, Bluetooth and NFC connectivities. It presents a strong integration of Cloud Computing and IoT, thus allowing to sense trash level and to communicate information in order to improve the waste management and its collection: both public and private entities can use information to optimize time and costs of recycling. The solution also allows to reduce traffic and pollution, and at the same time improving citizens' QoL. Citizens, through an application on their mobile devices, can detect the nearest Big Bucket where to depose a specific type of waste (community collection).

We are progressing towards the implementation and testing of this solution inside the #SmartME project in the city of Messina. One such scenario is the curbside collection, to test inside the same project, where citizens can control their own "smart bill". Moreover, we consider to design further typologies of smart dumpster for industrial environments, where the management of special wastes, both hazardous and not, is not trivial.

Acknowledgment. This work has been carried out in the framework of the CINI Smart Cities National Lab.

References

1. The ITU-T SG20 Study Group on IoT and its applications including smart cities and communities (SC&C). http://www.itu.int/en/ITU-T/studygroups/2013-2016
2. The European Waste Catalogue (EWC). http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014D0955&from=EN
3. The #SmartME crowdfounding project. http://smartme.unime.it/
4. Anagnostopoulos, T., Zaslavsy, A., Medvedev, A., Khoruzhnicov, S.: Top - k query based dynamic scheduling for IoT-enabled smart city waste collection. In: 16th IEEE International Conference on Mobile Data Management, vol. 2, pp. 50–55 (2015)
5. Anagnostopoulos, T., Zaslavsy, A., Medvedev, A.: Robust waste collection exploiting cost efficiency of IoT potentiality in smart cities. In: International Conference on Recent Advances in Internet of Things (RIoT), pp. 1–6 (2015)
6. Wang, J.Y., Cao, Y., Yu, G.P., Yuan, M.Z.: Research on application of IOT in domestic waste treatment and disposal. In: 11th World Congress on Intelligent Control and Automation (WCICA), pp. 4742–4745 (2014)
7. Chin, J., Callaghan, V.: Recyclable, eco-friendly, on-demand bin (ReDBin). In: International Conference on Intelligent Environments (IE), pp. 222–225 (2014)
8. The EU Framework 7 BURBA Project Report Summary. http://www.cordis.europa.eu/result/rcn/54
9. Bigbelly. http://bigbelly.com/
10. Enevo ONe. https://www.enevo.com/enevo-one/
11. Enevo. http://venturebeat.com/2014/11/19/trash-talk-enevo-equips-trash-bins-with-sensors-to-save-big-bucks/
12. Low-Power Wide-Area Technologies White Paper. https://www.lora-alliance.org/portals/0/documents/whitepapers/LoRa-Alliance-Whitepaper-LPWA-Technologies.pdf
13. The Omega Engineering LV3000/4000 series. http://it.omega.com/green/pdf/LV3000_LV4000.pdf
14. The First Sensor CT series. http://www.first-sensor.com/cms/upload/datasheets/DS_Standard-CTE-CTU-CTW9000CS_E_11594.pdf
15. Sotkis Intelligent Systems. http://www.sotkon.com/en/15/waste_management_systems
16. The Envac vacuum technology. http://www.envacgroup.com/using-envac
17. The Libelium Waspmote wireless sensor platform and smart cities sensor board. http://www.libelium.com/
18. IDENTIS WEEE. http://identisweee.net/
19. Arduino. http://www.arduino.org/
20. Merlino, G., Bruneo, D., Di Stefano, S., Longo, F., Puliafito, A.: Stack4Things: integrating IoT with OpenStack in a smart city context. In: Proceedings of the 2014 International Conference on Smart Computing Workshops (SMARTCOMP Workshops), pp. 21–28 (2014)
21. The OpenWRT Linux distribution. https://openwrt.org/
22. IoT resource management service for OpenStack clouds. https://github.com/MDSLab/iotronic
23. Stack4Things Framework. http://stack4things.unime.it/

Towards Distributed and Context-Aware Human-Centric Cyber-Physical Systems

Jose Garcia-Alonso[1(✉)], Javier Berrocal[1], Carlos Canal[2], and Juan M. Murillo[1]

[1] University of Extremadura, Badajoz, 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 increases, the interactions between the general population and Cyber-Physical Systems multiplies. Most of these interactions occur through people's smart devices. Thanks to the large number of sensor included on these devices and their capabilities to connect to other sensors they serve as a gateway to Cyber-Physical Systems. However, most of the capabilities of these devices are underutilized, since they are only used to upload the sensed information to centralized cloud servers. This paper presents the key challenges that must be faced to build distributed and context-aware human-centric Cyber-Physical Systems that take advantage of the capabilities of modern smart devices. In addition, the concept of Situational-Context is introduced as a possible solution addressing these challenges. Situational-Context is a new computational model that use smart devices to gather the virtual profiles of their owners. These profiles are combined and used to adapt and control the behaviour of Cyber-Physical Systems. This computational model could contribute to a new generation of distribute human-centric systems with a clear social orientation.

Keywords: Cyber-Physical Systems
Human-centric Cyber-Physical Systems · Internet of Things
Mobile computing · Situational-Context

1 Introduction

The increasing variety of internet-connected smart devices and their growing capabilities are facilitating the cyber-physical space to acquire special relevance in nowadays society. Putting together the capabilities offered by such huge amount of resources to perform tasks that could not be considered feasible so far is now becoming a practical challenge. The benefits go far beyond, providing functionalities that no single entity or system can achieve. This integrated behaviour is expected to start the next technological spin that will transform society, as it is currently known [15].

By no means can Cyber-Physical Systems (CPS) be considered a new concept. They have been present for many years in industrial facilities controlling

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 59–73, 2018.
https://doi.org/10.1007/978-3-319-72125-5_5

production chains through sensors and actuators, and more recently they have also been applied to other environments such as eHealth [20], automotive systems [39] or eldercare [2]. The next step is taking advantage of the high penetration of technology in current society to build a new generation of CPS [17] to be present in most aspects of daily life. For that, it is particularly interesting the massive involvement of humans in CPS.

This involvement is to a large extent managed through their smartphones. Such devices are the perfect sensors [12]. They are almost continuously active, always maintained by their owners and with the ability of measuring a broad variety of magnitudes, be it positioning, acceleration, direction, temperature, or heart rate. Consequently, sensors can provide data to compose a clear and reliable image of the context of their owners. However, the massive involvement of humans in CPS is not as perfect as expected. One of the main reasons for that is the architectural limitations.

Traditional CPS are commonly built based on a server-centric cloud architecture. With this architecture, the physical world data are provided by sensors and stored and processed by servers [7]. The data are processed to identify the need of performing actions over the physical world according to the CPS rules. The actions are translated to orders that are sent to actuators by the servers. Although this architecture has shown adequate for simple scenarios, it poses some limitations for the new generation of CPS. The higher the number of physical world entities involved in the CPS is, the higher storage and computation capacities are required on servers. Limitations that are much more serious if one considers that each person may be involved in several CPS (then, the sensed data would have to be uploaded to all these systems). A smartphone acting as a sensor simultaneously in several CPS will get its resources (especially battery) quickly drained. The natural consequence is that the owner of the smartphone will probably lose interest in the benefits of CPS and will leave it. An exodus of active members could make the CPS cease to have meaning.

This paper focuses on the key challenges that should be addressed to enable a new kind of CPS with significant involvement of humans. To address this challenge we propose the concept of Situational-Context, a new computational model that facilitates the distributed data storage and control. The principles behind that are simple: (1) instead of using smartphones as simply sensors, we can take advantage of all their storing and computation capabilities; (2) instead of uploading every data about their owners' context to servers, we leave them on the smartphones keeping them reachable and shareable for any CPS interested on them; (3) instead of computing every CPS rule on servers, we delegate on them the responsibility for the computation of the rules involving their owners' context. This distribute model allows people to be involved in several CPS, minimizing the data traffic between smartphones and servers and producing a sustainable consumptions. This work contributes to a new generation of massive, human-centric CPS with clear social orientation.

To describe the proposed model and the benefits it provides, this paper is organized as follows. Section 2 details the key challenges of building

human-centric CPS. In Sect. 3 the Situational-Context model is described emphasizing their effect on human-centric CPS and how can it be integrated with more traditional CPS. The most relevant related works are discussed in Sect. 4. Finally, Sect. 5 contains our conclusions and future works.

2 Key Challenges of Building Human-Centric CPS

Most CPS being built today follow a server-centric architecture. As Fig. 1 shows, this architecture is usually divided into two spaces. The physical-space, where the network of sensors and actuators are placed, and the cyber-space, where the processes controlling the CPS behaviour and the data used by them are placed.

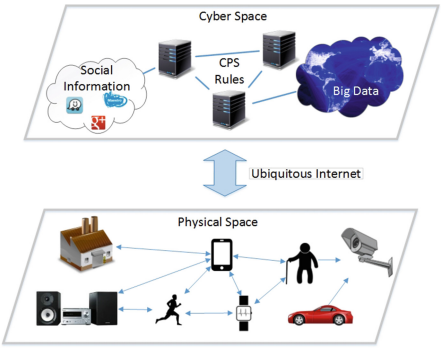


Fig. 1. Common architecture of CPS.

In this architecture, sensors obtain information from the physical world. Due to the ubiquitous internet connection, this information is propagated from sensors to the cloud servers where it will be processed. The information is then processed according to the CPS rules and commands are sent to the actuators, so they produce an effect on the physical world.

This architecture leads to very stable and reliable systems that provide suitable support for many environments. However, if we try to apply it to new contexts with massive involvement of people - i.e. social networks, smartcities, etc., it suffers limitations related to the amount of information that have to be processed and the way in which it is gathered.

To exemplify this limitations, we introduce a smart transportation scenario as running example. This kind of CPS, involving vehicles, are particularly affected by the systems communication [6]. The proposed system, through cameras and traffic lights, can control the congestion in order to avoid traffic jams. But it would be more useful to know, in real time, the route that the drivers are going to follow in order to foretell potential traffic problems and to accordingly control the traffic lights, or even to suggest drivers an alternative route. However, obtaining

such information from drivers and process it using a traditional architecture would be very expensive.

The constant advances in technology and, specially, the use of the smart devices for gathering the human context, has made the technological development of this kind of CPS feasible. In addition, the increasing storage and computing capabilities of these devices allows them not only to participate as mere sensors, but also to take a more active role in the architecture. However, the inclusion of this new role raises important challenges, which can be categorized in two sets, (1) human-related challenges, such as the ability of the CPS to adapt to the human context, the capacity of making complex inferences or the control of the privacy of the users' data, and (2) technological-related challenges, such as the control of the smart devices' resources, the provision of services from smart devices or the composition and orchestration of these services. Next, we describe these key challenges.

2.1 Human-Related Challenges

Based on the above, people are becoming the focus of more and more CPS. This increment in the relevance of people in CPS poses a set of challenges that are not addressed by traditional CPS. These are the most relevant people related challenges.

CPS Adapted to the Context of Their Human Users. Current CPS systems are usually very rigid and static. They analyse the information obtained by sensors and command specific actions to actuators upon a predefined set of rules. Therefore, they are not able to react differently to different users, environments or contextual situations [42]. This limits the personalization and human characterization of the CPS, reducing their acceptance by users. There are many situations, such as in everyday environments, in which CPS can provide a higher value if they can adapt to the context of the humans using them [28].

Currently, there are several approaches working on the development of context-aware CPS. These approaches propose different methods for monitoring and detecting the activities performed by users in a given context (such as eHealth [20] or supply chain [9]) to adapt the deployed CPS to the performed actions. However, in order to exploit all the benefits of human-centric CPS, these systems should be able to obtain more contextual information of the users, such as their preferences, interests, emotions or motivations [34].

For example, in the smart transportation scenario, it would be desirable that when a driver sits in her car, the CPS detects who has seated and adjust the seat and the dashboard luminosity to her preferences and establish in the car's GPS the route to follow to her destination.

Currently, as the general population begin to use a greater number of smart devices and wearables, the contextual information of the humans can be easily gathered. This information can be used to adapt the CPS behaviour to the context of their users. Therefore, in the previously defined example, the car

could interact with the user's smartphone in order to obtain her preferences and her destination, according to the next appointment established in her schedule.

Similarly, multiple benefits can be obtained if a CPS can be adapted not only to the contextual information of a single user, but if it can also take into account the contextual information of multiple users, providing a more consistent improvement. In the smart transportation example, once the car has detected the destination of the driver, it can detect that there will be a high amount of drivers that will also follow a similar route. Accordingly, an alternative route with a more fluid traffic would be recommended to the driver.

To summarize, there is a research challenge of being able to develop human context-aware CPS in which smart devices can actively adapt the system behaviour depending on the preferences of a specific user or to the context of a set of users.

Inference of Complex Sociological Data. Related to the use of contextual information, another challenge is the inference of complex sociological data. The complexity of the raw information obtained from users mostly depends on the sensors involved. Usually, this information is obtained from smartphones' sensors or other wearable technology, and involves relative simple data about the physical state of the user. However, much more complex and useful information can be inferred.

Social networks, such as Facebook or Twitter, and other IT companies, like Google, have demonstrated the profitability of the information inferred from their users. These organizations use the inferred sociological information to offer more personalized services, mainly advertisement. This information would be also highly useful to enhance the CPS systems, in order to be capable of reacting and adapting to it.

Many works have focused on defining systems for processing the large amount of information gathered [27]. These engines are mainly intended to be performed on servers with high computing and processing capabilities. However, this approach is not appropriate in situations in which the inferred information have to be reused in many different systems. For this, smart devices, that already obtain and maintain the contextual information, should also be able to infer the sociological information of their users and maintain the complete profile. Thus, an inference can be reused in different CPS.

In our smart transportation example, the usual route users follow from their homes to their jobs can be used to infer whether they have kids and their ages – if they usually make a stop at a school. This inferred information can be used both in a smart transportation CPS, in order to provide alternative routes to pass by the school, and also in a home automation CPS, for a parental control access to TV channels.

Currently there are no inference engines specifically adapted to the smartphones capabilities. For example, there are some studies, such as [14,40], analysing whether the already designed interferers can be deployed on smartphones. As result, these studies have identified that their deployment is feasible,

but their limited processing and memory capabilities lead to a diminution of the inferring speed.

Therefore, currently, there is a research challenge to develop interferers capable of exploiting the computing capabilities of smart devices in order to compose more comprehensive profiles of their users.

Privacy Issues Caused by Obtaining Contextual Information. Finally, as the number of smart devices and wearable technologies carried by people increases, more information is available. CPS could greatly benefit from it, but the privacy challenge becomes a serious concern when a system has several users and their data can be exposed to unauthorized parties [26].

Regarding our smart transportation scenario, the system could greatly benefit from constantly gathering the position of each user. This information could be used to control in real time the traffic and to redirect users to the most appropriate route at every moment. However, gathering this information implies that the system is able to know where each user is at any time, with the loss of privacy that this implies.

Currently, there are works focused in different aspects of the privacy and security of CPS. For example, some researches are focused on specific communication techniques [11,25], on detailing specific cryptographic mechanisms [30], or on defining systematic and complete architectures to manage the security in a CPS [26].

Nevertheless, the management of the users' information also depend on the policies established by each user on each individual system. In environments where the smart devices, participate in multiple CPS, the privacy management raises duplication issues, since the similar security policies have to be established in each environment. As it is highlighted in [16,31], there is a key challenge of approaches facilitating providing capabilities to easily establish security policies about who, how and when such information can be accessed.

2.2 Technological-Related Challenges

Directly related with the above described human-related challenges, there is a set of technological challenges that should be addressed during the design and development of human-centric CPS. These are the most relevant ones:

Battery Consumption of Monitoring Devices. As stated above, contextual information of its users can greatly benefit CPS performance. From a technological point of view, obtaining this information poses a significant challenge, mainly related to battery consumption. If the information is gathered by a battery powered device, the frequency at which the information can be obtained highly depend on the available battery.

Furthermore, once the information is obtained, in a traditional architecture, such data must be uploaded to a server for storing them and for executing the defined rules. However, uploading the data to the server also carries a significant

battery consumption. This expenses can be further increased for the devices participating in several CPS systems, since the same information should be uploaded to different CPS.

Therefore, CPS must be developed so that they take into account the battery life of the devices on their network [8]. This is not only a matter of physical limitation but also of user acceptance. In a human-centric CPS, users will automatically tend to reject the participation in any system that drains their smartphones batteries [38].

Current systems work around this challenge by gathering information every certain periods or only when specific situations occurs. For instance, most of the systems that obtain the users location do so whenever a significant change of position occurs, where significant can mean several hundred meters. This technique sacrifices precision and freshness of the information for battery savings. However, real time information is crucial in many situations. Works like [3] can help developers to analyse the battery consumption caused by CPS under different architectures.

Coming back to our smart transportation system, constantly uploading the real time position of a car may involve a reduction in the battery life of the device, but uploading this information only when there is a change of several hundred meters may also imply not correctly knowing the situation of a traffic problem. It would be more desirable to be able to access to fresh information on demand and under specific situations. Thus, the device's battery would be saved in those situations in which real time information is not required.

Therefore, currently, there is a technological challenge for obtaining fresh and updated information from devices under demand and without incurring on significant consumption of the battery life.

Providing Information Directly from Smart Devices. In a traditional CPS architecture, smart devices act as mere sensors and they are only used to gather information and send it to servers. However, this situation can be counterproductive for devices involved in several CPS and hinders the development of distributed CPS. It would be desirable that any device or server can directly access the information gathered and stored in the smart devices.

For example, the geolocalization of a driver can be used by the smart transportation system to improve the traffic congestion in a city, but also can be used by a home automation system to know when a person will arrive home and, thus, activate the air conditioning, the lights or the music. So, instead of constantly uploading the GPS information to each system's server, these devices should be capable of directly providing the information on demand.

Currently, there are approaches defining models and architectures for the deployment of lightweight services on smartphones, such as [13,29]. In [29], the authors propose a mobile-based hosting and serving architecture to eliminate the common cloud-based hosting of the media content being shared. Thus, the information collected and stored in the smartphones may be available to be used or shared in any CPS whenever required, without necessarily having to upload

it to each server. These studies only establish an architecture to deploy services on mobile devices. However, to the best of our knowledge, there is a research challenge for the definition of architectures for the deployment, provision and management of these services in a human-centric CPS.

Service Composition and Orchestration in Human-Centric CPS. If users devices start providing services that can be consumed by other entities of the CPS, not only servers but also other smart devices could make use of this information. This facilitates the development of distributed CPS, where smart devices are able to execute some CPS rules in order to orchestrate and compose complex services. In particular, human-centric CPS could greatly benefit from this orchestration of services, since they can adapt this orchestration to the context of their users.

This capability could be used to improve different aspects of the smart transportation CPS. For instance, if the smart device of a user detects an anomaly, like going slower than usual in a given spot, it could fire a CPS rule to directly communicate with nearby devices to verify if the anomaly is affecting them, too. If that is the case, the device would notify the traffic problem directly to a transportation control centre. This prevents the system to be constantly monitoring all drivers in order to detect traffic problems.

There are some works in the CPS and mobile fields focused in the orchestration of services deployed in devices. In [19], the authors present an orchestration framework for sensor-rich mobile applications. In [32], authors present a meta-model for modelling flexible and dynamic processes for the automation of workflows in CPS. Nevertheless, in a human-centric CPS, processes should be able to adapt to the human context [37]. Hence, there is an open research challenge to develop frameworks capable of orchestrate services adapted to the human context.

Large Amounts of Information to Be Processed. Finally, as more and more information is available, human-centric CPS need to process large amounts of information. This challenge is not inherent to CPS, but due to the advances in information technologies. Multiple research areas are involved in the processing of Big Data that could benefit the development of human-centric CPS.

Nevertheless with traditional methods, it is highly challenging to analyse the vast data volume generated by crowd sensing [41]. Meeting this challenge requires interdisciplinary approaches combining new architectures, novel algorithms and new processing environments [33]. Currently, some approaches, such as [18,36] are working on the deployment of simulation models and novel algorithms in cloud environments to provide higher processing and better accessibility resources. However, these techniques imply that the sensed information should be uploaded to the cloud environment, with the consequent waste of resources.

A research challenge for this kind of systems would be the use of smart devices as the highly capable computational entities they are. If the different devices

process their own information, the need for a centralized service processing huge amounts of information would be mitigated.

2.3 Addressing the Challenges

All the above challenges should be addressed to build truly human-centric CPS. To address them, existing techniques could be added to traditional CPS. However, more efficient results could be obtained by applying a paradigm shift in the way CPS have been traditionally built. This paradigm shift involves completely changing the role played by smart devices and wearable technology in this kind of systems.

The penetration of these devices, their ever increasing computing and sensing capabilities, and the fact that they are constantly carried everywhere by their owners makes them the perfect candidate to be a more active element in almost any human-centric CPS.

Below, a new computational model for personal yet collaborative contextualization of smart devices enabling distributed CPS is introduced. In this model, smart devices gather, infer, and store the contextual and sociological information of their users and execute the CPS rules delegated to them, adapting them to their users' context, exposing complex services or initiating actions in other devices. Thus, new human-centric CPS minimizing the data traffic and the battery consumption can be developed.

3 Situational-Context

The Situational-Context [4] can be defined 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 CPS. The Situational-Context is form by the resulting context of locally composing the virtual profiles of the different entities (things and people) involved in a particular situation. For a meaningful composition of these profiles, we consider that they contain, at least, the following information:

- A *Basic Profile* containing the dated raw information with the entity's status, the relationships with other devices and its history. This profile can be seen as a timeline with the changes and interactions that happened to the entity.
- *Social Profile.* This profile contains the results of high level inferences performed over the Basic Profile.
- The *Goals* detailing the status of the environment desired by the entity. These Goals can also be deducted from the Basic and Social Profiles.
- *T*he Skills or capabilities that an entity have to make decisions and perform actions capable of modifying the environment and aimed at achieving Goals.

The result of composing the virtual profiles of the involved entities is not only the combined information of all entities. It contains 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 the Goals based on the present Skills should be identified. These strategies will guide the prediction of the interactions that must emerge from the context.

Furthermore, the Situational-Context is a dynamic abstraction of the combined profiles and, therefore, evolves 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.

Figure 2 shows the Situational-Context for controlling the temperature of a room. It contains a first configuration (C1) combining the virtual profiles of a thermostat and the smartphone of a person that is in the room. The smartphone defines the Goal to have a comfort temperature and the thermostat has a Skill to control the temperature of the room. When a new user with the same Goal in her profile enters the room, the situation change, a new configuration (C2) is computed and the strategies required to achieve the combined Goals are identified. Then, the interactions required for setting the adequate comfort temperature emerge from this context.

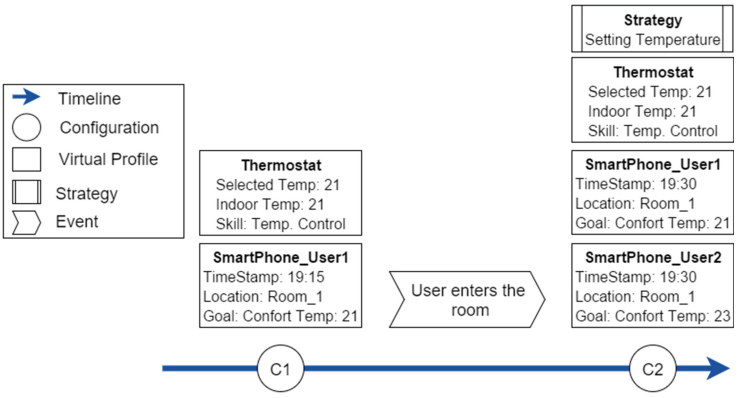


Fig. 2. Excerpt of a Situational-Context.

By applying the Situational-Context model to the development of human-centric CPS we can address the human-related challenges detailed in the previous section. The use of the virtual profiles of all the users involved in a particular situation would allow developers to build distributed CPS that transparently adapt to the preferences and needs of their users. As detailed above, these virtual profiles would include the social profiles of the users, obtained as a result of applying inferences over the basic profiles. In addition, when the profiles of all

the entities involved in a situation are combined, higher level inferences are performed to better adapt the CPS behaviour. Additionally, by keeping the virtual profiles in the smart devices, instead of sending them to each CPS in which the users are involved, the privacy is improved and its management is simplified.

Similarly, the technological-related challenges can be addressed by an implementation of the Situational-Context. However, due to the early stage of development in which the concept of the Situation-Context is, not implementation is available yet. Nevertheless, the authors of this work have been working in the past years in the implementation of several concepts like People as a Service [10], the Internet of People [24] or the analysis of the consumption of mobile applications under different architectures [3]. A more detailed description of these works can be found in Sects. 2 and 4 of this paper. By using the technical advances presented in those works, an implementation of the Situational-Context model can be created that address all the above described technological-related challenges.

4 Related Work

Currently there are a number of works exploiting the capabilities of smart devices in order to provide them a more active role in CPS.

VITA [12] proposes an architecture for mobile devices in order to facilitate the development and management of mobile crowdsensing apps (for collecting and aggregating sensing data). Also, this system supports the allocation of computational and human tasks to different smartphones in run-time. This system is a step forward in the use of smart devices as an active element in the human-centric CPS, but it still delegates a lot of the responsibility on the servers.

In [23], the authors indicate that mobile phones can be used to form wireless sensor networks in order to sense various information, such as to identify people in crowded areas. In these networks the autonomy of the mobile phone is critical, so that they have optimized the OLSR routing protocol to increase it. This protocol facilitates the achievement of the challenge associated with the battery consumption in human-centric CPS.

In the Ambient Intelligence paradigm, some works use multi-agents systems to gather the users' context and to react in a proactive and autonomously way to it [35]. However, the constant access to virtual profiles stored on mobile devices can lead to a quickly drain of their battery, which is a crucial aspect for them and for the success of mobile applications [22]. The Situational-Context model intends that these profiles are accessed only once to be composed. Therefore, since the different inference rules and strategies can be directly executed using the combined profile, the battery consumption is reduced.

The Fog Computing [5] extends the Cloud Computing paradigm to the edge of the network, using one or a collaborative multitude of end-users devices or near-user edge devices to store information or to execute different processes. Thus, reducing the latency and the communication overhead.

The nimBees notification platform [1] provides a mobile a API that can be included in any iOS or Android application. This API allows mobile developers

to easily gather contextual information about the users of their applications. This information is processed by the nimBees platform inference engine to create the sociological profile of the device owners. This profile is keep on the device and used to segment the push notifications sent to the application. This enables a segmentation technology based on the sociological profiles of the owners without compromising their privacy.

PeaaS [10] aims to use smartphones as the virtual representation of their owner. By using smartphones in this way, the sociological profiles of their owners are transparently gathered and provided to other systems. These profiles can be used to better adapt human-centric CPS to the needs and requirements of their users and, since PeaaS revolves around maintaining the users private information in their own devices, privacy is greatly improved over traditional systems.

Social Devices [21] aims to exploit the capabilities of these devices to better acknowledge the social connections between their owners. These social connections can then be translated to the cyber-space of the CPS to help implement human-centric CPS. Mainly, Social Devices will help deal with the management of the multiple devices owned by users and with the coordination of the composed services provided by those devices. It will also add to human-centric CPS the capabilities to adapt to the needs of large amount of people by taking into account their social relationships.

The combination of both previous paradigms has resulted in the Internet of People manifesto [24]. This manifesto seeks to bring human-centric enhancements to existing technologies. These works have been used as the basis for developing the computational model presented in this paper.

Of course, a distributed system is not always the most optimal solution. When CPS are designed, they have to be analysed and, depending on their requirements and characteristics, a server-centric solution could be more efficient than a distributed solution. In [3], we propose a conceptual framework that developers can use to identify which architecture is more efficient or, even, is the best option is a combination of both.

5 Conclusion

CPS with massive involvement of humans requires devices capable of adapting their behaviour to their users' context and of collaborating between them in order to better meet their users' needs. This requires to provide smart devices a more active role in CPS.

In this paper, we have presented an ongoing work detailing a computational model for distributed human-centric CPS allowing smart devices to store, infer and provide information on their owners. The use of this model enables the development of collaborative and contextualized human centric CPS. With the additional advantages of reducing the consumption of the smart devices' resources and increasing the user privacy. The benefit provided by the model, however, are only significant for CPS with certain characteristics. Its focus on adapting the system to the context of the users, makes this architecture to be especially oriented to CPS with massive human involvement.

As further work, we are planning to apply the defined model in real-world scenarios. Concretely, we are implementing it for an automotive scenario. This experiment will be used to measure the reduction in the consumption of the smart devices' resources.

Acknowledgments. This work was partially supported by the Spanish Ministry of Science and Innovation (projects TIN2014-53986-REDT, TIN2015-67083-R and TIN2015-69957-R), by the Department of Economy and Infrastructure of the Government of Extremadura (GR15098), and by the European Regional Development Fund.

References

1. nimBees Platform. http://nimbees.com/
2. Berrocal, J., Garcia-Alonso, J., Murillo, J., Canal, C.: Rich contextual information for monitoring the elderly in an early stage of cognitive impairment. Pervasive Mob. Comput. (2016)
3. Berrocal, J., Garcia-Alonso, J., Vicente-Chicote, C., Hernandez, J., Mikkonen, T., Canal, C., Murillo, J.: Early analysis of resource consumption patterns in mobile applications. Pervasive Mob. Comput. (2016)
4. 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
5. Bonomi, F., Milito, R., 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 2012), pp. 13–16. ACM, New York (2012)
6. Bradley, J.M., Atkins, E.M.: Optimization and control of cyber-physical vehicle systems. Sensors **15**(9), 23020–23049 (2015)
7. Chen, N., Xiao, C., Pu, F., Wang, X., Wang, C., Wang, Z., Gong, J.: Cyber-physical geographical information service-enabled control of diverse in-situ sensors. Sensors **15**(2), 2565–2592 (2015)
8. Frank, R., Mouton, M., Engel, T.: Towards collaborative traffic sensing using mobile phones (poster). In: 2012 IEEE Vehicular Networking Conference (VNC), pp. 115–120, November 2012
9. Frazzon, E.M., Hartmann, J., Makuschewitz, T., Scholz-Reiter, B.: Towards socio-cyber-physical systems in production networks. In: Forty Sixth CIRP Conference on Manufacturing Systems, vol. 7, pp. 49–54 (2013)
10. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. Softw. IEEE **31**(2), 48–53 (2014)
11. Hancke, G., Markantonakis, K., Mayes, K.: Security challenges for user-oriented RFID applications within the "internet of things". J. Internet Technol. **11**(3), 307 (2010)
12. Hu, X., Chu, T., Chan, H., Leung, V.: Vita: a crowdsensing-oriented mobile cyber-physical system. IEEE Trans. Emerg. Top. Comput. **1**(1), 148–165 (2013)
13. Jansen, M.: Evaluation of an architecture for providing mobile web services. Int. J. Adv. Internet Technol. **6**(1), 32–41 (2013)
14. Kazakov, Y., Klinov, P.: Experimenting with ELK reasoner on android. In: Bail, S., Glimm, B., Gonçalves, R.S., Jiménez-Ruiz, E., Kazakov, Y., Matentzoglu, N., Parsia, B. (eds.) CEUR Workshop Proceedings, vol. 1015, pp. 68–74 (2013)

15. Kim, K.D., Kumar, P.: Cyber-physical systems: a perspective at the centennial. Proc. IEEE **100**(Special Centennial Issue), 1287–1308 (2012)
16. Kobsa, A.: Privacy-enhanced personalization. In: Tsihrintzis, G., Virvou, M., Howlett, R., Jain, L. (eds.) New Directions in Intelligent Interactive Multimedia. Studies in Computational Intelligence, vol. 142, p. 31. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68127-4_3
17. Lee, E.A.: The past, present and future of cyber-physical systems: a focus on models. Sensors **15**(3), 4837–4869 (2015)
18. Lee, J., Lapira, E., Bagheri, B., Kao, H.A.: Recent advances and trends in predictive manufacturing systems in big data environment. Manuf. Lett. **1**(1), 38–41 (2013). http://www.sciencedirect.com/science/article/pii/
19. Lee, Y., Min, C., Ju, Y., Kang, S., Rhee, Y., Song, J.: An active resource orchestration framework for pan-scale, sensor-rich environments. IEEE Trans. Mob. Comput. **13**(3), 596–610 (2014)
20. Li, T., Cao, J., Liang, J., Zheng, J.: Towards context-aware medical cyber-physical systems: design methodology and a case study. Cyber Phys. Syst. 1–19 (2014)
21. Mäkitalo, N., Pääkkö, J., Raatikainen, M., Myllärniemi, V., Aaltonen, T., Leppänen, T., Männistö, T., Mikkonen, T.: Social devices: collaborative co-located interactions in a mobile cloud. In: Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (MUM 2012), pp. 10:1–10:10. ACM, New York (2012). http://doi.acm.org/10.1145/2406367.2406380
22. Merlo, A., Migliardi, M., Caviglione, L.: A survey on energy-aware security mechanisms. Pervasive Mob. Comput. **24**, 77–90 (2015). http://www.sciencedirect.com/science/article/pii/S1574119215000929. Special Issue on Secure Ubiquitous Computing
23. Meseguer, R., Molina, C., Ochoa, S., Santos, R.: Reducing energy consumption in human-centric wireless sensor networks. In: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1473–1478, October 2012
24. Miranda, J., Mäkitalo, 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)
25. Ning, H., Liu, H.: Advances in intrusion detection system for WLAN. Adv. Internet Things **1**(3), 51–54 (2011)
26. Ning, H., Liu, H.: Cyber-physical-social based security architecture for future internet of things. Adv. Internet Things **2**(1), 1–7 (2012)
27. Perera, C., Zaslavsky, A.B., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. J. **16**(1), 414–454 (2013)
28. Plödereder, E., Grunske, L., Schneider, E., Ull, D. (eds.): 44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 22–26 September 2014 in Stuttgart, Deutschland. LNI, vol. 232. GI (2014)
29. Raatikainen, M., Mikkonen, T., Myllärniemi, V., Mäkitalo, N., Männistö, T., Savolainen, J.: Mobile content as a service a blueprint for a vendor-neutral cloud of mobile devices. Softw. IEEE **29**(4), 28–32 (2012)
30. Roman, R., Alcaraz, C., Lopez, J., Sklavos, N.: Key management systems for sensor networks in the context of the internet of things. Comput. Electr. Eng. **37**(2), 147–159 (2011)
31. Roman, R., Zhou, J., Lopez, J.: On the features and challenges of security and privacy in distributed internet of things. Comput. Netw. **57**(10), 2266–2279 (2013)
32. Seiger, R., Keller, C., Niebling, F., Schlegel, T.: Modelling complex and flexible processes for smart cyber-physical environments. J. Comput. Sci. (2014)

33. Sharma, A.B., Ivančić, F., Niculescu-Mizil, A., Chen, H., Jiang, G.: Modeling and analytics for cyber-physical systems in the age of big data. SIGMETRICS Perform. Eval. Rev. **41**(4), 74–77 (2014). http://doi.acm.org/10.1145/2627534.2627558
34. Sánchez-Escribano, M., Sanz, R.: Emotions and the engineering of adaptiveness in complex systems. Procedia Comput. Sci. **28**, 473–480 (2014). 2014 Conference on Systems Engineering Research
35. Sorici, A., Picard, G., Boissier, O., Florea, A.: Multi-agent based flexible deployment of context management in ambient intelligence applications. In: Demazeau, Y., Decker, K.S., Bajo Pérez, J., de la Prieta, F. (eds.) PAAMS 2015. LNCS (LNAI), vol. 9086, pp. 225–239. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18944-4_19
36. Wan, J., Zhang, D., Zhao, S., Yang, L., Lloret, J.: Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. Commun. Mag. IEEE **52**(8), 106–113 (2014)
37. Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context integration for smart workflows. In: Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), pp. 239–242, March 2008
38. Wilke, C., Richly, S., Gotz, S., Piechnick, C., Assmann, U.: Energy consumption and efficiency in mobile applications: a user feedback study. In: 2013 IEEE International Conference on Green Computing and Communications (GreenCom), and IEEE Internet of Things (iThings), and IEEE Cyber, Physical and Social Computing (CPSCom), pp. 134–141 (2013)
39. Work, D., Bayen, A., Jacobson, Q.: Automotive cyber physical systems in the context of human mobility. In: Proceedings of the National Workshop on High-Confidence Automotive Cyber-Physical Systems, Troy (2008)
40. Yus, R., Bobed, C., Esteban, G., Bobillo, F., Mena, E.: Android goes semantic: Dl reasoners on smartphones. In: Bail, S., Glimm, B., Gonçalves, R.S., Jiménez-Ruiz, E., Kazakov, Y., Matentzoglu, N., Parsia, B. (eds.) CEUR Workshop Proceedings, vol. 1015, pp. 46–52 (2013)
41. Zhang, Y., Chen, M., Mao, S., Hu, L., Leung, V.: CAP: community activity prediction based on big data analysis. Netw. IEEE **28**(4), 52–57 (2014)
42. Zhuge, H.: Semantic linking through spaces for cyber-physical-socio intelligence: a methodology. Artif. Intell. **175**(5–6), 988–1019 (2011)

Application Development and Deployment for IoT Devices

Farshad Ahmadighohandizi(✉) and Kari Systä(✉)

Department of Pervasive Computing, Tampere University of Technology,
Korkeakoulunkatu 10, 33720 Tampere, Finland
{farshad.ahmadighohandizi,kari.systa}@tut.fi

Abstract. The current IoT systems are based on simple sensors that collect data to a cloud and actuators controlled by applications in the cloud. When the devices become more powerful they will eventually become programmable and host applications. In this paper we present an application framework, development tool and execution platform for such applications. The guiding principle in designing the workflow design has been Continuous Delivery (CD) and DevOps, and our paper also provides a comparison of DevOps in the cloud and DevOps for the IoT.

Keywords: IoT · Continuous Delivery · DevOps
Resource Registry and Discovery · Dynamic update
Programmable things

1 Introduction

Although the current IoT devices are very resource constrained and only designed to collect data to a backend server for further processing [4] we assume that the IoT devices of the future will be programmable and capable of hosting applications. This will open possibilities for new types of applications and dynamic updates to the systems. This is a huge contrast to traditional embedded systems where the devices are pre-programmed for a certain application. We foresee that the change to programmable devices opens a new world of intelligent systems where devices collaborate to achieve the goals of the user.

When remote devices are programmable, the management and operation of them becomes a challenge. IoT devices are typically headless, i.e., they do not include any UI. This makes remote device management a central feature of an IoT system. Remote device management may include setting and updating configuration of the devices, monitoring the state of the devices, and over-the-air updates of devices' firmware [6].

The development and deployment of cloud-based internet services are increasingly done in a continuous manner. This approach is called Continuous Deployment [9,14]. If the management and operations of the applications and the underlying infrastructure are done independently from the application developers, the overall coordination is compromised and the development is slowed down.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 74–85, 2018.
https://doi.org/10.1007/978-3-319-72125-5_6

DevOps [15] tries to address this by removing the boundary between development and operation. We see that dynamic programming of IoT devices has similar needs to continuous deployment of IoT devices.

Programming of future IoT systems and production of cloud-based applications have several similarities. In this paper we describe an approach where the development and deployment of IoT applications are done with an approach inspired by principles of DevOps. The cornerstones of the approach are: (1) remote management of IoT resources including applications, and (2) an integrated tool for development, deployment and operation.

In addition to introducing the concept, the paper present a proof-of-concept of the application framework and development tool. The framework runs currently on Raspberry PI[1] and Intel Edison[2] hardware and is build on top of Nodejs[3]. Similar system could be built on other HW and SW platforms, too. The development tool is based on Ace[4] technology and runs in standard browsers.

The rest of this paper is organized as follows. Section 2 describes the background and prior work on the area. Section 3 introduces the concept, its implementation, and its components and their roles. Section 4 compares the approach with Continuous Deployment of cloud-based applications. There are many similarities but also interesting differences between CD of cloud-based Internet services and that of IoT applications. Finally, Sect. 6 gives conclusions and points out some future work.

2 Background

This section briefly describes DevOps and CD and also presents our prior work on a new kind of DevOps tool for cloud-based development of Web applications.

2.1 Continuous Deployment and DevOps

Frequent deployment of new versions of software to production is not possible if teams responsible for development and IT-operation are separated from each other. DevOps is a mind-set that addresses the issue of siloed teams. Both developers and operation people can benefit substantially from an integrated DevOps tool that meets the requirements of all tasks. Developers are able to develop the application while operations personnel can get the executable, deploy it, and manage the operations of the applications.

2.2 An Integrated Development Tool for DevOps

This research is utilized the earlier cloud-related research at Department of Pervasive computing in Tampere university of Technology. One research direction

[1] https://www.raspberrypi.org.
[2] http://www.intel.com/content/www/us/en/do-it-yourself/edison.html.
[3] https://nodejs.org/.
[4] https://ace.c9.io/.

has been cloud- and browser-based development tools, for example CoRED [20]. In ITEA2 project EASI-CLOUDS[5] in 2013–2014, TUT's contribution was to research integrated development tool that supports the DevOps mind set [2]. This proof-of-concept was based on CoRED. We also implemented a proof-of-concept of the tool using the Web and cloud as the infrastructure. Inspired by principles of DevOps, users of the tool can develop web applications, deploy them into the available cloud providers, and also manage the operations of the deployed applications with the same tool. Provisioning the required environments and deployment and further management of applications are done using two components called COAPS [22] and ACCORDS [26].

3 Programmable Platform for IoT Devices

The platform presented in this paper has three main components. The first component is an application framework that is supported by a browser-based Integrated Development Environment (IDE). The IDE also supports deployment of applications and management of their operations. The second component is a runtime environment which turns an IoT device into an application server that can run IoT applications. The last component is a Resource Registry (RR) mechanism to discover available IoT Resources including devices and applications. All these components support our application framework that defines the life-cycle and execution models of the applications.

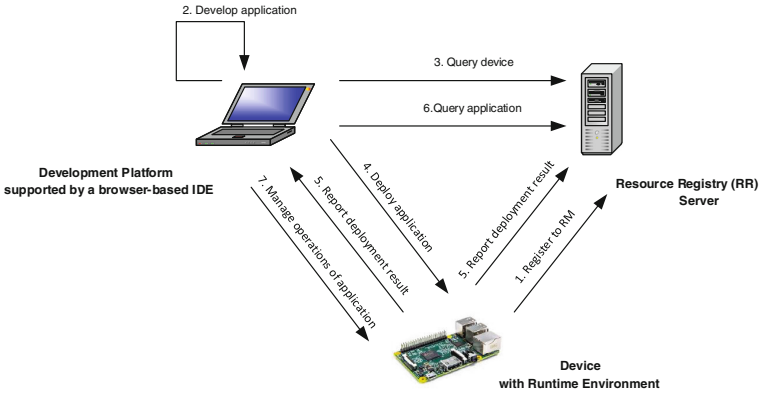


Fig. 1. Basic scenario for application development, deployment, and management

Figure 1 illustrates the different components and operation of our platform and tool. The steps are as the following:

1. When a new device is installed, it is registered with the Resource Registry (RR) Server. This server keeps track of all known devices.

[5] https://itea3.org/project/easi-clouds.html.

2. A new application is developed or found among pre-existing applications.
3. Suitable device(s) are discovered from RR. See Sect. 3.3 for details.
4. The application is deployed to the discovered device via Application Management APIs provided by Runtime Environment.
5. The deployment result is returned to the development tool, and in a case of successful deployment the installed application is registered to the RR, too.
6. The developer tool or any other entity can search applications from RR, and
7. The tool can be used for monitoring and management of the application.

3.1 Application Framework

The framework provides a standard structure for the application to fit in our IoT platform. It provides developers with functionalities (in the form of functions) that need to be either filled in with application specific code or called by the application. As our implementation is based on JavaScript virtual machine, the framework is based on a JavaScript object representing the IoT application.

When the user creates a new application, the IDE generates a template that the developer need to complete by filling in the missing code. The functions that need to be filled in with code are:

- *task*. This is the function that gets called periodically at certain intervals or only once. The frequency can be controlled with function *setTaskInterval(repeat, interval)*. The parameter *repeat* defines whether the task function should be executed periodically or only once.
- *initialize*. This function gets called before task function starts. It is mainly used for initialization purposes like establishing a connection or initializing variables.
- *terminate*. This function gets called before the application stops. It is mainly used for cleaning things up like killing a connection.

The execution model of our applications is event-based and asynchronous. Usually, the application code is executed by calling function *task* periodically. These asynchronous events can be interleaved with other events like completion of I/O operations. Thus the platform provides additional means to control the order of events.

In addition to periodically called *task* functions the applications can also provide services including RESTful APIs to access these services. These APIs can be called from anywhere in the network. We call these RESTful APIs as *Application Interfaces (AI)* and we use the Open API Specification standard[6] as an approach to work with AIs. The IDE includes an API editor[7] for defining and managing of AI specifications. The library of developed AIs is available to developers for development of applications that either offer or use these APIs. After choosing which AIs the application should implement, a skeleton code is added to the application.

[6] https://github.com/OAI/OpenAPI-Specification.
[7] http://swagger.io/swagger-editor/.

The tool packs the developed application to one package containing the following files:

- *main.js*. A file that defines the IoT application source code.
- *Package.josn*. A file that contains metadata of the application including application's name, description, version, list of dependencies, etc.
- *liquidiot.json*. A file that is used for discovery and bookkeeping mechanism (see Subsect. 3.3). It declares the required device capabilities and provided AIs of the application.
- *resource*. A folder for application resources like images and sounds.

3.2 Runtime Environment and Management Interface

The runtime is based on Node.js technology and turns an IoT device into an application server that can host and run IoT applications. Runtime environment offers a set of RESTful APIs for over-the-air deployment and remote management of applications. Further information about application management APIs can be found from http://farshadahmadi.github.io/.

3.3 Resource Registry Service

A fundamental requirement of IoT resource management is a mechanism to discover available resources in an IoT system and their capabilities and properties including means to access them [7]. In our platform, resources are categorized into four types: (1) *applications*, (2) *devices*, (3) *Device Capabilities* and (4) *Application Interfaces*. In our proof of concept implementation, these resources are registered to and managed by a centralized Resource Registry (RR) service. RR also provides discovery mechanism to discover the aforementioned available resources.

4 CD in Cloud vs IoT

Usually CD is used in application development for virtualized cloud infrastructures. However in this research we have applied CD to development and deployment of IoT devices. While investigating CD opportunities in IoT we have found interesting similarities and differences between continuous deployment of cloud-based internet services and IoT applications.

4.1 Virtual Host in the Cloud vs Physical Device

When applications are deployed in the cloud environment, new virtual hosts can be created. Since virtual hosts are offered as cloud services, the number of hosts is not limited. In the case of IoT, edge devices are not offered as a cloud service. Therefore, the number of hosts is fixed to the existing devices and resources available in those devices. CD in IoT must consider this limitation while creating new execution environments.

In CD systems for cloud it is common that while the host is created, the PaaS technology builds a fresh execution environment to match the needs of the installed application. In the IoT case, the devices may be very heterogeneous but at the same time the core functionality of the device cannot be changed. For example, deploying an application to measure temperature to a device without required sensor is not feasible. So, besides setting up the execution environment, the developers also needs to find suitable devices and adapt to them. Consequently, the deployment pipeline needs discovery and bookkeeping mechanisms to find the suitable target devices.

4.2 Deployment Strategies

One of the main deployment questions is whether the new version should immediately replace the old version or should the old version kept active, too. Various deployment patterns address this challenge differently. This Subsection investigates these deployment patters and their advantages and disadvantages. The analysis also compares cloud and IoT environment in this respect.

In *in-place deployment* the new version of the application replaces the old one. So, during deployment the application is unavailable resulting in a period of downtime. If just the application is updated, the downtime may be short. However, if the execution environment changes, for example, a new version of operating system or platform components is needed, those changes need to be applied to the host before deployment of the application. This causes longer downtime and contains risks and rollback to old version is hard. If the environment changes significantly, it is often easier to build up the execution environment from scratch.

Often the application is served by several hosting environments and a load balancer controls the distribution of application traffic. In these cases the *rolling deployment* pattern [3] can be used. While in-place deployment is being done once at a time on each server, the host under deployment is detached from the load balancer and attached again only after successful deployment. Compared with simultaneous in-place deployment to all environments, sequential deployment takes more time. However, the advantage is zero downtime since there are always some servers running either the old or new version of the application. Also, the process can be interrupted if the deployment fails – leaving old hosts operational. The other pros and cons are the same as for in-place deployment.

Sometimes it is necessary to ensure that the new version really works before the old version can be removed. The *blue-green deployment* pattern [10] has been designed for that purpose. In this pattern the old and still used environment is called blue and a new environment, called green, will be provisioned to host the new version of the application. Once the software is stable in the new host, the old and new environment URLs will be swapped by changing Domain Name System (DNS) configuration. The first advantage is zero down time. Secondly, rolling back to old version can be done by swapping the URLs back instead of redeployment. Similarly to rolling deployment, the risks and issues related to changes in execution environment can be mitigated. The disadvantage is that if

the old environments includes services like persistent data, the synchronization of that data is complicated.

For current IoT devices only the in-place deployment is obvious to implement. The other approaches assume alternative hosts that are usually based on virtualization. A limited version of blue-green deployment can be implemented, if the IoT device can host two versions of the same application. In low-resource devices hosting of two versions of the application is not always possible, which means that the deployment will cause a short downtime. Then, if the new version is found to be faulty, a rollback to old version is needed. However, if the system can host several versions and instances of the application, many benefits of blue-green deployment can be reached. Our proof-of-concept is based on rather powerful hardware and thus supports several applications in the same device. The deployment creates a separate execution environment. Thus, we could implement a simple blue-green deployment of the apps.

Since the IoT devices do not support functionalities similar to modern PaaS technologies, possibilities to provision a new execution environment are limited. Our current proof of concept creates new execution environments by spawning new Nodejs processes. By this approach, compared with approaches using virtualization, a limited management of execution environments can be implemented. This means that the benefits of the advanced deployment patters cannot be fully utilized. However, we see that with appearance of virtualization in IoT, management of execution environment will progress dramatically in the future.

4.3 Role of Staging

Staging environment is an exact replica of the production environment. When the software is stable in the staging, it will be taken to live production. IoT devices are tightly connected to a physical world so it is impossible to create an exact replica for staging. However, a lot of final testing should be done in a simulator, and a simulator of an IoT device can act as a staging environment. We have not implemented this in our proof of concept but it would be an interesting topic for future investigations.

Another option to implement staging is to use the blue-green deployment. The blue-green deployment pattern provides a great opportunity to host the staging environment in the target device and test the application there. If the application works properly in the staging, the staging will be swapped by the production.

4.4 Bulk Deployment to Production

While blue-green deployment works for deployment into a single device, it is not scalable for bulk deployment to production. Suppose that one application is deployed to a thousand devices using blue green deployment. Now, if the application fails, all the thousand staging environments will fail. Although this does not cause a serious problem since failures happened in staging not production,

redeployment should be done again to all devices. Use of Canary deployment [21] pattern can solve this problem.

Canaries were used in coal mining since their early sign of distress to toxic gases was an indicator of danger to miners [25]. Similarly, if the deployment is first done to only a small subset of devices, they can play the role of canaries. If the canary deployment fails, the rest of devices are not affected. If the software is stable in "canary devices", it indicates that deployment to the rest of devices is safe.

5 Related Work

Our work is closely related to recent trends of Edge and Fog Computing. Edge Computing is a paradigm where the computing is moved from central servers to remote ends of the network. Usually this means end user devices but in the context of IoT it means sensors and actuators. If computation happen at the proximity of IoT devices, many problems including slow response time, huge unnecessary network bandwidth, storage for large quantity of data, and security will be mitigated [23]. In our platform, we are moving computation right up to the IoT devices through making them programmable dynamically. In Fog Computing [5] the storage, computing capacity and applications are distributed to several nodes in the network and the applications operate in a collaborative way. Our system has similar features since our applications in devices provide AIs that allow applications to collaborate.

Application programming for IoT devices usually follows the practices of embedded programming and the low power IoT devices use lightweight embedded operating systems like TinyOs [13] or Contiki [8]. These operating systems are based on specific programming languages (e.g., nesC [11] for TinyOs) and communication middleware (e.g., Active Message [4] for TinyOs). The fundamental limitation of these systems is poor separation of the application from platform code, and they do not support separately installed applications. The basic assumption of our framework is that applications can be developed and installed separately from the platform code. The proof of concept is based on JavaScript programming language and virtual machine provided by the Node.js technology. Moreover, the runtime environment spawns a new process to run a new application providing a level of isolating for different applications.

ELIoT [24] is a development platform for internet-connected smart devices. ELIoT code is compiled into a platform independent bytecode which is then executed in heterogeneous devices using ELIoT VM. An ELIoT-enabled device has the ability of dynamically spawning processes so that a new capability can be added over-the-air to the device by executing the related fragment of code in the created process. Our platform uses VM provided by Node.js technology and runs applications in spawned Node.js processes. In ELIoT inter-process communication is done through asynchronous message passing. Broadcast communication, where a process needs to send a message to multiple other processes, is used to implement the discovery of available resources. Our discovery mechanism

is centralized where available resources are queried from a centralized server. While ELIoT network-wide communication is done through REST interfaces of processes, in our platform both network-wide and inter-process communication are through REST Application Interfaces provided by applications.

Actinium [18] is designed for programming networked embedded systems where each IoT device only offers a simple set of functionality like sensing and actuation through RESTFul interfaces. Actinium applications are then realized through scripts running in Actinium Runtime Environment in the cloud. In our platform, applications run in the devices and are not in the cloud only communicating with the devices. Applications in both our platform and Actinium provide RESTFul APIs for communicating with other applications. Mashups of applications is possible in both platforms by using APIs provided by applications. However, in Actinium application logic runs outside of the networked devices; in our platform, the logic can run inside the network.

Laukkarinen proposes "embedded cloud" [19], a solution to distribute application logic among heterogeneous resource-constrained IoT devices. Application logic is developed with Process Description Language (PDL) that allows platform independent process creation and is distributed among devices as PDL processes with help of a Distributed Middleware (MiDiWa). MiDiWa abstracts the heterogeneity of IoT devices and provides a homogenized service interface for PDL processes to be both executed and shared as services. In our platform, the application logic can be distributed through Application Interfaces provided by IoT applications.

Aaltonen et al. [1] propose an action-oriented programming model to create multi-device programs. Capabilities are installed on devices by running software resources that implement certain interfaces. Under certain conditions, triggers notify actions that utilize device capabilities to coordinate joint behaviour between several devices. This differs from [12] and Node-RED[8] that offer orchestration of devices through the notion of "physical mashups" in which the logic runs outside of the devices. Our platform allows coordination of applications through the Application Interfaces they implement.

Datta and Bonnet [7] propose a resource discovery framework for IoT similar to our registry and discovery framework. The difference is the query mechanism of resources. The resources are indexed and then queried by a search engine. Our platform uses a library (csstomongo.js[9]) to turn discovery selectors into mongodb[10] queries to query the resources.

IBM Bluemix[11] developed a cloud-hosted Internet of Things service called Watson IoT Platform[12] for collecting data from connected devices. Devices are registered to and managed by a centralized IoT Platform Device Management server. Watson platform provides REST APIs to the applications that want to

[8] http://nodered.org.
[9] https://gist.github.com/ahn/7d82017bef2f5641c7e7.
[10] https://www.mongodb.com/.
[11] http://www.ibm.com/cloud-computing/bluemix/.
[12] http://www.ibm.com/cloud-computing/bluemix/internet-of-things/.

consume the device data. Our platform, as well as collecting the device data in a remote backend, allows dynamic programming of devices. While both platforms use centralized registry, our resource registry service enjoys a discovery system that enables querying of available resources including devices and applications.

6 Conclusion

We presented a programmable platform for IoT devices. The platform consists of three components. The first component is an application framework for development of IoT applications. The framework is supported by a browser-based IDE. Besides applications, Application Interfaces can also be developed in the same tool. Application Interfaces are RESTful APIs through which the applications can provide services to each other. The second component is a Runtime Environment which turns an IoT device into an application server so that it can host and run IoT applications and offer an interface for application deployment and management of operations. The last component is a Resource Registry (RR) server. The fundamental requirement of IoT resource management is a mechanism to discover available resources in an IoT system and their capabilities and properties including means to access them. In our platform the system has four kinds of resources: (1) available devices, (2) device capabilities, (3) installed applications, and (4) Application Interfaces. RR keeps track of all available resources.

After presenting our framework and its proof of concept implementation we compared the enabled application development and deployment facilities to continuous deployment practice in virtualized clouds. We found out that CD practice needs to take some considerations in the case of IoT. CD should consider the limited number of hosts offered by the available devices in an IoT platform while creating new hosts for application deployment. The deployment pipeline must also support discovery of available resources including the target device since they have certain abilities and therefore can only be used for special purposes. The different deployment patterns were also investigated in the context of IoT devices. While in-place deployment is an obvious choice for low-resource devices, powerful devices can leverage some benefits of the blue-green deployment pattern. Bulk deployment into production can follow canary deployment pattern in which the result (success of failure) of blue-green deployment into a small subset of devices determines whether to deploy to the rest of devices or not.

As in the future work we want to address multi-device programming so that applications can utilize several devices. Currently we are investigating three different approaches. The first approach is graphical data wiring [12] similar to NodeRED. The second approach is orchestration of REST services [16]. The last approach is action oriented programming model [1]. Other goals include adding of the test environment using cloud technologies and the staging environment following blue-green deployment pattern. Our current proof-of-concept assumes 2-way HTTP requests and addressability which is not well-suited with practical applications for example due to Network Address Translation (NAT). So, improving protocol support (in particular MQTT [17]) is another future work.

We also plan to experiment additional types of devices with different capabilities in our platform. So far, our development platform has been used and tested mainly inside the research group, establishing a code camp helps us receive feedback from programmers outside the development team.

References

1. Aaltonen, T., Myllärniemi, V., Raatikainen, M., Mäkitalo, N., Pääkkö, J.: An action-oriented programming model for pervasive computing in a device cloud. In: 2013 20th Asia-Pacific Software Engineering Conference (APSEC), vol. 1, pp. 467–475, December 2013
2. Ahmadighohandizi, F., Systä, K.: ICDO: Integrated cloud-based development tool for DevOps. In: SPLST (2015)
3. Amazon Web Services: Deployment policies and settings. http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html. Accessed 30 June 2016
4. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
5. Bonomi, F., Milito, R., 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 2012), New York, pp. 13–16. ACM (2012)
6. Boswarthick, D., Elloumi, O., Hersent, O.: M2M Communications: A Systems Approach. Wiley, Hoboken (2012)
7. Datta, S.K., Bonnet, C.: Search engine based resource discovery framework for internet of things. In: 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), pp. 83–85, October 2015
8. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462, November 2004
9. Fitz, T.: Continuous deployment, February 2009. http://timothyfitz.com/2009/02/08/continuous-deployment/. Accessed 30 June 2016
10. Fowler, M.: Bluegreendeployment, March 2010. http://martinfowler.com/bliki/BlueGreenDeployment.html. Accessed 30 June 2016
11. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC language: a holistic approach to networked embedded systems. In: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI 2003), New York, pp. 1–11. ACM (2003)
12. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In: Internet of Things (IOT) 2010, pp. 1–8 (2010)
13. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), New York, pp. 93–104. ACM (2000)
14. Humble, J.: Continuous delivery vs continuous deployment, August 2010. https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment. Accessed 30 June 2016
15. Humble, J., Molesky, J.: Why enterprises must adopt devops to enable continuous delivery. IT J. **24**(8), 6–12 (2011)

16. Hylli, O., Lahtinen, S., Ruokonen, A., Systä, K.: Service composition for end-users. Acta Cybern. **21**(3), 383–399 (2014)
17. ISO/IEC: ISO/IEC 20922:2016 information technology - message queuing telemetry transport (MQTT) v3.1.1
18. Kovatsch, M., Lanter, M., Duquennoy, S.: Actinium: a RESTful runtime container for scriptable internet of things applications. In: 2012 3rd International Conference on the Internet of Things (IOT), pp. 135–142, October 2012
19. Laukkarinen, T., Suhonen, J., Hännikäinen, M.: An embedded cloud design for internet-of-things. Int. J. Distrib. Sens. Netw. (2013)
20. Lautamäki, J., Nieminen, A., Koskinen, J., Aho, T., Mikkonen, T., Englund, M.: Cored: browser-based collaborative real-time editor for Java web applications. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW 2012), New York, pp. 1307–1316. ACM (2012)
21. Sato, D.: Canaryrelease, June 2014. http://martinfowler.com/bliki/CanaryRelease.html. Accessed 30 June 2016
22. Sellami, M., Yangui, S., Mohamed, M., Tata, S.: PaaS-independent provisioning and management of applications in the cloud. In: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), pp. 693–700, June 2013
23. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **PP**(99), 1 (2016)
24. Sivieri, A., Mottola, L., Cugola, G.: Building internet of things software with ELIoT. Comput. Commun. **89**, 141–153 (2016)
25. Stovell, P.: Canary deployments, July 2014. http://docs.octopusdeploy.com/display/OD/Canary+deployments. Accessed 30 June 2016
26. Yangui, S., Marshall, I.J., Laisne, J.P., Tata, S.: CompatibleOne: the open source cloud broker. J. Grid Comput. **12**(1), 93–109 (2014)

Cloud Adoption and Migration (CloudWays)

Preface of CloudWays 2016

Cloud computing has recently been the focus of attention both as academic research and industrial initiatives. From a business point of view, organizations can benefit from the on-demand and pay-per-use model offered by cloud services rather than an upfront purchase of costly and over-provisioned infrastructure. From a technological perspective, the scalability, interoperability, and efficient (de-)allocation of resources through cloud services can enable a smooth execution of organizational operations.

Regardless of the benefits of cloud computing, many organizations still rely on business-critical applications – so-called legacy systems – developed over a long period of time using traditional development methods. In spite of maintainability issues, (on-premise) legacy systems are still crucial as they support core business processes that cannot simply be replaced. Therefore, migrating legacy systems towards cloud-based platforms allows organizations to leverage their existing systems deployed (over publicly available resources) as scalable cloud services.

This second edition of the workshop – the 2nd International Workshop on Cloud Adoption and Migration (CloudWays 2016) – was held in Vienna, Austria on 5 September 2016 as an ESOCC satellite event. The first edition was held in September 2015 in Taormina, Italy, also as a satellite event of ESOCC. The workshop's goals were to bring together cloud migration experts from both academia and industry; to promote discussions and collaboration amongst participants; to help disseminate novel cloud migration practices and solutions; and to identify future cloud migration challenges and dimensions.

In this second edition, four full papers and one short paper were accepted for presentation during the workshop, out of a total of nine submissions.

The first paper "Cloud Migration Architecture and Pricing - Mapping a Licensing Business Model for Software Vendors to a SaaS Business Model" by Frank Fowley and Claus Pahl targeted the link between architecture and cost in cloud application migration, pointing to the challenges particularly for software vendors to migrate the system in parallel to the business model in order to adapt to cloud constraints.

The second paper "A DMN-based Approach for Dynamic Deployment Modelling of Cloud Applications" by Frank Griesinger, Daniel Seybold, Jörg Domaschka, Kyriakos Kritikos and Robert Woitsch has looked at the dynamic nature of cloud deployments and proposed a decision support layer to assemble an abstract deployment model.

The third paper "Cloud Migration Methodologies: Preliminary Findings" by Mahdi Fahmideh Gholami, Farhad Daneshgar and Fethi Rabhi.reviewed a number of existing migration methodologies in a systematic format. A criteria-based framework allows the extraction of benefits and limitations of each methodology.

The fourth paper "Workflow Skeletons: Improving Scientific Workflow Execution through Service Migration" by Tino Fleuren investigated specifically service migration in the context of scientific workflows. Workflow skeleton are the mechanism to manage migration.

The final (short) paper "Consumer-Driven API Testing with Performance Contracts" by Johannes Stähli, Sebastian Lang, Fabian Kajzar and Christian Zirpins reports on quality aspects in service migration. API testing based on performance contracts is the mechanism to assert quality requirements.

In addition to the presentation of the accepted papers, an invited talk titled "Semantics and Patterns to support MultiCloud Applications' Portability and Cloud Services Orchestration and Composition" was jointly organized with participants of the PEACE in PATTWORLD workshop focusing on the challenges and perspectives of patterns and semantics in cloud-based systems. The presentation was given by Prof. Beniamino Di Martino from the Second University of Naples, Italy.

We take this opportunity to thank all authors, members of the Program Committee and workshop attendees, whose participation was invaluable to the success of the event. We also acknowledge the support provided by The Irish Centre for Cloud Computing & Commerce (IC4) and the Free University of Bozen-Bolzano (UniBZ).

Claus Pahl
Nabor Mendonça
Pooyan Jamshidi

Organization

Program Committee

Aakash Ahmad	IT University of Copenhagen, Denmark
Vasilios Andrikopoulos	University of Stuttgart, Germany
Thais Batista	Federal University of Rio Grande do Norte, Brazil
William Campbell	Birmingham City University, UK
Fei Cao	University of Central Missouri, USA
Santiago Gomez	University of Stuttgart, Germany
Sören Frey	Daimler TSS, Germany
Wilhelm (Willi) Hasselbring	Kiel University, Germany
Abbas Heydarnoori	Sharif University of Technology, Iran
Pooyan Jamshidi (Co-chair)	Imperial College London, UK
Ali Khajeh-Hosseini	RightScale, Inc., UK
Xiaodong Liu	Napier University, Edinburgh, UK
Theo Lynn	Dublin City University, Ireland
Paulo Henrique Maia	State University of Ceará, Brazil
Nabor Mendonça (Co-chair)	University of Fortaleza, Brazil
Claus Pahl (Co-chair)	Free University of Bozen-Bolzano, Italy
Dana Petcu	West University of Timisoara, Romania
Américo Sampaio	University of Fortaleza, Brazil
Amir Sharifloo	University of Duisburg-Essen, Germany

Cloud Migration Architecture and Pricing – Mapping a Licensing Business Model for Software Vendors to a SaaS Business Model

Frank Fowley[1] and Claus Pahl[2(✉)]

[1] IC4, Dublin City University, Dublin, Ireland
[2] Free University of Bozen-Bolzano, Bolzano, Italy
Claus.Pahl@unibz.it

Abstract. Cloud migration is about moving an on-premise software system into the cloud. Many approaches exist that describe the technical migration analysis and the architectural migration. Equally, cloud cost models have been investigated. What we aim to investigate here is to link architecture and software utilisation to costing and business models. We specifically look at software vendors that use the cloud to provide their solutions to customers. They might face the challenges to migrate an in-house developed and provided product onto a cloud IaaS or PaaS platform, while also mapping a licensing model onto a cloud monetisation model. We provide here an experience report. This is based on experience with five migration case studies. We discuss the migration process under consideration of cost aspects, covering both income and expenses in the cloud in relation to the cloud delivery model chosen. We focus on one of the case studies to illustrate the concepts and observations.

Keywords: Cloud migration · Cloud cost models · Monetisation
Architecture migration · Independent software vendor · Cloud native

1 Introduction

The adoption of cloud computing and in particular the software as a service (SaaS) model causes problems for existing on-premises applications and independent software vendor (ISV) business models.

ISVs are often forced by external factors to adopt the new model. The threats and opportunities involved in this step shall be investigated. The demand for on-premises applications is shifting to the SaaS model. In order to maintain revenue growth and profitability, ISVs have to provide applications deployable on cloud services from a technical perspective. But more importantly, ISVs generally need to go beyond offering just a cloud-enabled application to offering their application value as SaaS, which dramatically changes the business model in addition to the technical challenges [1,2,4,5].

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 91–103, 2018.
https://doi.org/10.1007/978-3-319-72125-5_7

The technical benefits are well discussed [8,11,12]. Being able to scale up or down application infrastructure to meet quality requirements and enable reliable consumption of a product is a key benefit. Cloud migration research has studied the cloud on-boarding in quite some detail [13]. Processes and techniques are proposed. For instance, pattern-based migration processes are suggested to organise and manage the architectural migration. Tools have been provided by many cloud service provider to migrate for instance data using data loaders.

In [21], the top 10 challenges for start-ups are summarised, that also reflect the concerns for migrations by software vendors inexperienced in cloud technologies, particularly if the cloud context is a novel environment.

- Thriving in Technology Uncertainty, i.e., developing technologically innovative products that require cutting-edge development tools and techniques such as cloud technologies is the top challenge.
- Acquiring First Paying Customer, Delivering Customer Value and Defining Minimum Viable Product are other concerns within the top-10. These all relate to defining a payment model for a quality product and making it viable considering the costs for providing cloud SaaS as well.

A significant change is that the up-front license revenue model is replaced by over-time subscription revenue [14–17]. This is generally a disruptive process. Revenue in the cloud builds up more slowly because of typical SaaS pricing models. With traditional on-premises licensing, each organization that buys a copy of the software provide a substantial income to the ISV at one moment. With the subscription model common in the SaaS, the customers will only pay a small amount per user per month. This is prone to fluctuate.

An additional complication is that this needs to be planned for as an incremental process. For most ISVs the shift from on-premise to SaaS delivery will be incremental. To some extent this can also help as when this incremental process is coupled with license sales, it might be easier to maintain profit stability, but would need to be aligned with any incremental technical migration. Effective customer retention, which can be achieved through the cloud, the subscription revenue model can be made to work. Another benefit often cited is the opportunity to grow business.

The key observation is that sound models do not exist at present that allow ISVs to map their existing business model to the cloud. Instead of trying to provide a concrete calculation model, which is not possible due to the complexity of pricing models [5,6], we report on observations from five case study migrations. The cover ISVs with software product in the insurance, banking, food sectors as well as two generic business solutions for image processing and registry data management. The image processing case will be singled out to illustrate the observations later on in a concrete setting.

The paper is organised as follows. The next section introduces the link between technical architectures and costing models. Section 3 reviews the state-of-the-art. Section discusses the monetisation concerns from a technical architecture perspective. Section 5 shows the different costing implications of IaaS and PaaS-based architectures. Section 6 then goes into more detail about PaaS

costing models. Section 7 looks at the SaaS perspective of charging customers for the product in the cloud. Section 8 summarise a few concrete concerns learnt from the use case. Section 9 finally discussed current changes in cloud charging models that might impact this in the future towards more predictability, before we end with some conclusions.

2 Linking Cloud Architecture and Services to License Management and Billing

The key concern for the ISV is the product monetisation. A number of technology requirement emerge for a successful SaaS monetisation [10]:

- Platform as a service (PaaS) solutions provide development tools and support necessary to create and deliver software (as a service). PaaS generally includes support for service provisioning to the customer, which includes metering, monitoring and auditing tool to determine and analyse who accesses and utilizes the provided resources.
- License management allows a service provider to enforce, analyse and manage use parameters related to how the service is licensed or provisioned. This includes support for typical cloud and also more complex provisioning scenarios (i.e., beyond per user/per month or unlimited enterprise usage for the latter category).
- Billing takes over from monitoring and metering to implement the subscription model and automate the collection of fees. Billing includes the ability to automate metering, pricing and billing for products, bundles and configurations. It looks after handling of recurring payments associated with user subscriptions.

The technical concerns to feed into the monetisation process are the following:

- Accounting and Billing: Automate orders, provisioning, entitlements, billing, and ongoing customer management
- Access Control: Controlling access to service features, which could be time limited, role specific, or value based
- Metering: Measure service usage at a granular level for the purposes of compliance, billing, and product intelligence
- Automation of accounting: Remove manual processes entirely, such as those around trial conversions when the policy is not to auto-convert

Figure 1 provides the architecture including the relevant income-related components for the SaaS provider.

3 Context and Related Work

Both academia and industry provide migration methodologies and tools. These cover anything from cloud readiness and benefit analysis to the development of migration plans to execution methods and tools.

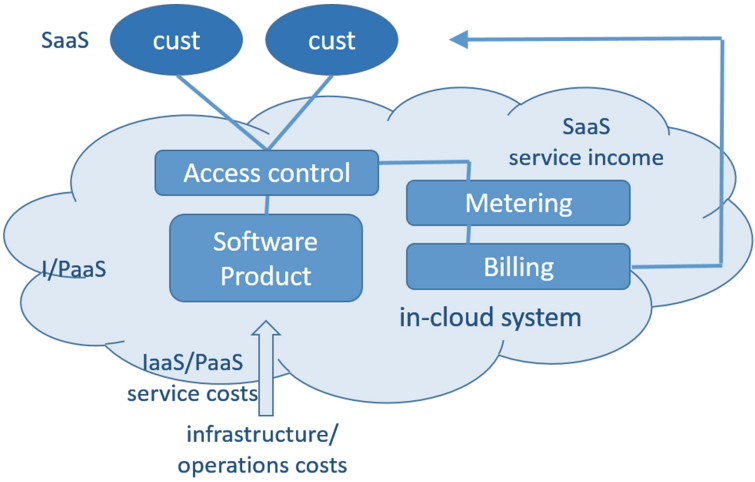


Fig. 1. A PaaS architecture with monitoring, metering and billing components

Industry guides such as white papers by D. Chappell[1] provide frameworks of concerns, but lacks a concrete calculation method. This is still pretty much a what-to-do, but not a how-to-do coverage. A survey of academic research on migration is presented by Jamshidi et al. [3].

In addition to the lack of practical advice, costs and their mutual dependencies with architectural decisions are not well addressed [9]. There are academic paper proposing solutions to calculate profit for cloud services.

Wang et al. [20] proposed an algorithmic solution to optimize data center net profit with deadline-dependent scheduling.

4 Cloud Migration – Joint Architecture and Costing Concerns

4.1 Pre-migration – Licensing Model of ISVs

The traditional monetisation approach for software vendors involves a licensing model, which needs to be off-set against costs for development and operation.

An independent software vendor (ISV) is an organization specializing in making or selling software, designed for mass or niche markets. This is in contrast to software developed for in-house use only within an organization or software designed or adapted for a single, specific customer.

Independent Software Vendors typically operate a licensing model.

[1] http://www.davidchappell.com/writing/white_papers/How_SaaS_Changes_an_ISVs_Business--Chappell_v1.0.pdf
.

4.2 Architecture Migration and Monetisation

The actual migration planning and execution then needs to consider a number of models reflecting the different concerns involved:

- Architecture: source and target architecture needs to be considered together with planned changes in functional or non-functional properties.
- Cost model (expenses): the projected expenses in the cloud need to be determine, which includes basic infrastructure and platform costs, but also all additional features for external access and networking, internal quality management and possibly development and testing costs.
- Revenue model (income): the projected income based on a selected pay-per-use or subscription model (or a combination).

These together result in a monetisation model for the cloud-provided software product, which is illustrated in Fig. 2.

From a cloud architecture perspective the following concerns need to be considered:

- Components: the components of the provided software product - componentisation is important to consider their mapping onto different cloud services that might be differently charged.
- Usage metering and Billing: applies to the services used, but also the services provided.

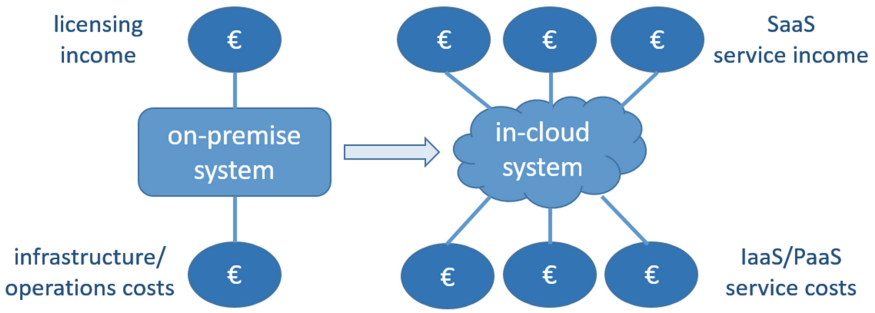


Fig. 2. Monetisation architecture - mapped from on-premise to cloud

5 Architecture Migration – IaaS to PaaS Architecture Mapping Towards Cloud-Native

An important consideration is the full adoption of the cloud as a development platform. By shifting development to a PaaS cloud, allows ISVs to avoid capital expenditure and map investment in developer facilities to revenue potential.

Another important concern is the staged migration. Through basic virtualisation a simple VM-based IaaS solution might emerge. The ultimate objective

would be to move from VMs to cloud native applications at platform level. This of course changes the expenses model, resulting in something more fine-granular.

Assume a traditional stacked application with Application, Middleware, DBMS and Disk Storage support that runs in an on-premise setting, with the aim of providing this as a service in the cloud. A step-wise migration into cloud could now happen as follows:

1. As a first, simple solution, this application could be packaged into VMs:
   (a) situation: license fees for components of the application would occur as usual.
   (b) business problem: scaling out, i.e., adding more VMs (for technical of other reasons) means adding more license fees for every replicated component.
   (c) technical problem: multiple copies of data storage that are not in sync, causing integrity problems.
2. A second step could refactor and extract storage, i.e., use a virtual data-as-a-service solution for storage needs:
   (a) this alleviates the technical integrity problem resulting from different copies of data.
3. A further step could package the whole DBMS into single virtual machine:
   (a) this alleviates the business license fee problem for the DBMS and simplifies data management.
   (b) but business problems remain: other license fees do still occur multiple times.
4. Finally, the ISV could consider to fully move to PaaS services (e.g., using PaaS services for data management and other concerns).
   (a) apart from solving key technical problems, this alleviates as far as possible the licensing fees issue.

Following this process results in what is called cloud-native, and can be characterised through the following properties:

scalable/elastic, clusterable, multi-tenant, pay-per-use, self-service

This is assumed to have better scalability characteristics. It also, as already said, allows better licensing and cost management.

This of course only addresses the cost side for the ISV. Technically, this goes towards what microservices principles by being independently deployable using fully automated deployment machinery. This requires orchestration support for topologies that goes beyond provisioning only. And, furthermore, predicting the cost for a fully cloud-native solution is the challenge. The suggested refactoring towards a cloud-native solution alleviates licensing problems, but the ultimate costs till remain unclear. Finding a model how to set fees for the SaaS usage is also a challenge.

6 Architecture Migration and PaaS Deployment Cost Calculation

6.1 Architecture Selection and Deployment

Selection criteria for an ISV for a platform to provide the software product from include:

- functional requirements, supported by programming language databases and database support and also procedures for version control, testing and deploying code (tools such as Eclipse, Git or Jenkins supported by many PaaS).
- fees, where many providers offer monthly subscription fees, but extra features will incur extra cost: examples here are scalability, access (IP endpoint, network bandwidth) or monitoring and advanced self-management.

6.2 Cost and Revenue Prediction/Calculation

What is needed is a calculation of the cloud costs versus the possible revenue result from the SaaS delivery model. Note that we mainly discuss PaaS here, although similar assumptions can be made for IaaS. PaaS-level costs need to cover both development and deployment, which needs to be balanced against SaaS-level income. The following steps need to be taken to determine some reliable figures:

- Predict costs based on cloud costs (from basic virtualisation to cloud-native) for an assumed usage (workload) of the SaaS application provided to customers.
- Determine income (by estimating usage and choosing suitable fees levels) under consideration of, firstly, the current licensing model and resulting revenue and, secondly, under consideration of achieving this through a staged, incremental migration.

6.3 Pricing Models

In order to systematise this calculation, we list here common factors of a PaaS or IaaS pricing model:

- Region – slightly different rates might apply per region (which is relevant of for instance data location regulations apply).
- Replication – is a mechanism to deal to avoid down-time and increase reliability. Sample configurations could be:
  - Local Redundant – a number of copies of data, all in the same data-centre, in the region of the storage account across different Fault or Upgrade domains (physical rack units managed as a unit)
  - Zone Redundant – a number of copies of data, all in different data-centres, which has slightly less throughput than Local redundancy

  - Geo Redundant – a number of copies of data, all in different data-centres, with a back-up, separate multiple saves in a specific secondary region to allow to recover from Region failure
  - Read-Only Geo Redundant – is the same as geo redundancy with read access to secondary data

  All replication operations are done asynchronously.
- Size – depends on actual amount of Gbytes stored.
- Transactions – Read/Write Blob Operations are counted.
- Data Transfer – is measured. Some sample costings are as follows:
  - Data Ingress Network Data Transfer is free.
  - Data Egress Network Data Transfer is free if in the same region.
  - Data Egress Data Transfer between regions or out of a region is charged.

Based on these general cost factors, IaaS or PaaS providers usually create account types with specific pricing models for each of the factors[2]:

- A General Purpose Storage Account: has tables, blobs and queues. Itemized prices for specific redundancy schemes (default Local Redundancy) could be:
  - Storage: Euro 0.0202 per GB per month (Zone Redundant: Euro 0.0253) (Geo Redundant: Euro 0.0405)
  - Transactions: per 10,000 Euro 0.003 (Zone Redundant: Same) (Geo Redundant: Same)
  - Egress from Region: Euro 0.0734 per GB per month (after free 5 GB) (Zone Redundant: Euro 0.1164) (Geo Redundant: Euro 0.1526)

  A Blob Storage Account for block blobs would have the following settings:
  - Hot access tier for frequently accessed data. Cold access has lower storage costs, but higher access and transactions costs.
  - A Cool access tier for archive and back-up, i.e., not frequently accessed data, is an option, but needs to be available immediately when needed. This is assumed to be the case for image blobs, if needed.
  - The Cool access is cheaper, but might require a separate storage account for tables and blobs, which in turn means that the same shared access keys cannot be used.
  - Cool Prices for Local Redundancy as the default could be:
    * Storage: Euro 0.0126 per GB per month (Geo Redundant: Euro 0.0253)
    * Transactions: Put, List blob per 10,000 Euro 0.0843 (Geo Redundant: Euro 0.1687)
    * Data Read: Euro 0.0084 per GB (Geo Redundant: Euro 0.0084)
    * Data Write: Euro 0.0021 per GB (Geo Redundant: Euro 0.0042)
    * This also has the General Purpose costs for Egress from Region: Euro 0.0734 per GB per month (Geo Redundant: Euro 0.1526)

[2] We have taken the concrete figures from recent (May 2016) Microsoft Azure pricing models.

Relevant pricing models focus primarily on storage in GB and transactions (read/write). What this illustration shows is that a very good understanding of the assumed or predicted workload of a provided SaaS application is needed on order to calculate the costs for hosting the SaaS application for example in a PaaS cloud. This requires quantified workload (in terms of GB of data stored/transferred and number of transactions), but also certainty about other quality concerns (availability expectations, failover strategy). So, what effectively the aim in the calculation is, (i) as input the number of storage units and total size as input and (ii) as output the costs as output calculate over a number of years with predicted growth and for different replication options.

A further complication is that pricing models between different platform provider are difficult to compare due to the fragmentation and itemisation of the factors determining the pricing of a service. Consequently, a formalised mapping between possible licensing models of a SaaS application and the incurred costs of hosting this on a PaaS offer cannot exist, but we demonstrate a process here to carry of this calculation in a concrete case. After an analysis of the PaaS pricing models for hosting an application, we look at licensing and pricing models for offering a PaaS-hosted application as a SaaS solution.

7 Princing Model Migration – Cloud SaaS Billing and Pricing Models

SaaS pricing models typically bill clients using one of the following two metrics[3]:

- The number of users accessing the application.
- The volume of resources consumed.

Different account types such as pay-per-user or pay-as-you-go exist based on these individual factors.

7.1 Pay-Per-User SaaS Pricing Model

Pay-per-user is a simple and popular SaaS pricing strategy. With this model, a separate cost is incurred for each user of a SaaS application. This is consequently similar to the traditional payment for each copy of software that is put on a device. The advantage compared to traditional pricing is the SaaS ubiquity, which is essentially available on most devices today. This does not cause separate fees for the different device types. SaaS billing happens periodically for all users, e.g., monthly. SaaS software entitlements can be managed using an identity management system.

There are variations of the pay-per-user SaaS pricing model. One example is the pay-per-multiple-user pricing model. Here a separate cost is incurred for a defined number of users, creating different tiers for certain number ranges such as 1...9, 10...50 etc. A further option is to bundle an increasing number of features within each successive tier.

[3] https://apprenda.com/library/software-on-demand/saas-billing-pricing-models/.

7.2 Pay-As-You-Go (Utility) SaaS Pricing Model

The second popular SaaS pricing model is the pay-as-you-go model. Here, the charges are based on the number of users and the amount of resources consumed during a defined period of time (typically volume of storage or CPU usage). This makes the prediction or expenses (for the user) or income (for the provider) generally difficult, but the pay-as-you-go model is beneficial for users because only the actual volume of resources consumed is charged, as opposed to a flat rate model that is not fully utilised. But this is a user benefit, but not for the ISV for whom this unpredictability is a difficulty.

8 Use Case Study

We now report on the calculation process in a concrete use case. We carried out this investigation for a concrete company providing a business administration solution. This solution is migrated into the cloud to respond to changing business requirements, particularly:

- flexibility, allowing different access forms
- expansion, requiring to facilitate new customers in new markets

The challenge is a high-volume data storage and processing need. The earlier pricing samples reflect this storage/processing need already.

8.1 Performance Experimentation and Prediction

The first step was to determine performance statistics based on some assumed workloads:

- standard operations were scaled up to estimated peak loads for the application,
- response times and CPU/memory consumption were measured on different resource configurations to determine a link between resources needed and QoS provided,
- resulting in the identification suitable configurations that would maintain SLA compliance for all customers.

This experimentation can include the comparison of similar platform services within the services of oe provider, but also between different providers where the comparison is possible. For the case study, we have compared Azure with AWS pricing with similar settings. While both options were initially considered, the fact that significant parts of the system were developed in .NET lead to Azure being chosen, which in this case was not a cost-driven decision.

8.2 Income Prediction and Income Model Determination

A further collection of input data for the calculation includes:

- to determine the number of users,
- to determine the expected consumption/load of a typical user (based on the performance experimentation),
- to determine licensing model options using different account types.

This the definition of a pricing model maps costs per typical user, taken from the experiments, into a licensing model.

9 A Changing Environment

New provisioning and payment models moving away from pay-per-hour models towards payment by business cycles seem to emerge recently in PaaS. They link the SaaS provisioning costs for the ISV.

AWS lambda functions are a new feature. AWS Lambda is a compute service where you code is uploaded and the Lambda service runs the code using AWS infrastructure. The upload code is used to create a so-called Lambda function. The AWS Lambda service then handles provisioning and managing the servers to run the code. Interesting in this your model is the charging approach. A user (such an ISV) is charged based on the number of requests for the ISV's functions and the time the ISV code executes, i.e., is based on (a) Requests and (b) Duration. This model is a first approach to link ISV PaaS costs to the ISV SaaS income (through the SaaS utilisation).

Google has announced a similar Cloud Functions model as a lightweight, event-based, asynchronous compute solution that allows the creation of small functions that respond to cloud events without the need to manage a server or a runtime environment. Cloud Functions are written in Javascript and execute in a managed Node.js environment on Google Cloud Platform. Events from the Cloud Storage and Google Cloud Pub/Sub can trigger these functions asynchronously or HTTP invocation is used for synchronous execution.

10 Conclusions

A holistic perspective on costing and architecture within a migration scenario does not exist. There are metering and billing solutions, be that as part of commercial products or as part of research prototypes. There are also migration frameworks targetting architectural transformation and planning the process. An investigation linking architectural decisions and the impact on costing is therefore important and has not been looked at apart from case studies [24].

We have reviewed the key components of such a holistic framework. This investigation has resulted in a process to calculate the costs for hosting a SaaS application on a PaaS platform and use this to determine a SaaS licensing model. As a generic, formalised model cannot exist due to the differences in factors and

account types between the PaaS providers, our aim was to identify the factors influencing this calculation and to exemplify this through a concrete example.

Note that selection of services as a consumer is also a selection process that compares functionality, quality and costs, but this stage has been neglected [18].

From a technical perspective, we also aim to investigate container technology and a microservices style architecture [23] to see their impact in the context of cloud-native architectures [7,19,22].

Acknowledgement. This work was partly supported by IC4 (the Irish Centre for Cloud Computing and Commerce), funded by EI and the IDA.

References

1. Arshad, S., Ullah, S., Khan, S.A., Awan, M.D., Khayal, M.: A survey of cloud computing variable pricing models. In: International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) (2015)
2. Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., Ahmad, I.: Cloud computing pricing models: a survey. Int. J. Grid Distrib. Comput. **6**(5), 93–106 (2013)
3. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)
4. Brangewitz, S., Hoof, S.: Economic aspects of service composition: price negotiations and quality investments. In: Aiello, M., Johnsen, E.B., Dustdar, S., Georgievski, I. (eds.) ESOCC 2016. LNCS, vol. 9846, pp. 201–215. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44482-6_13
5. Samimi, P., Patel, A.: Review of pricing models for grid and cloud computing. In: IEEE Symposium on Computers and Informatics (ISCI), pp. 634–639. IEEE (2011)
6. Chang, V., Wills, G., De Roure, D.: A review of cloud business models and sustainability. In: 2010 IEEE 3rd International Conference on Cloud Computing, pp. 43–50. IEEE (2010)
7. Aldawood, S., Fowley, F., Pahl, C., Taibi, D., Liu, X.: A coordination-based brokerage architecture for multi-cloud resource markets. In: 4th International Conference on Future Internet of Things and Cloud Workshops. IEEE (2016)
8. Pahl, C., Xiong, H.: Migration to PaaS clouds - migration process and architectural concerns. In: IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA 2013 (2013)
9. Walterbusch, M., Martens, B., Teuteberg, F.: A decision model for the evaluation and selection of cloud computing services: a first step towards a more sustainable perspective. Int. J. Inf. Technol. Decis. Making **14**(02), 253–285 (2015)
10. Garrison, G., Wakefield, R.L., Kim, S.: The effects of IT capabilities and delivery model on cloud computing success and firm performance for cloud supported processes and operations. Int. J. Inf. Manag. **35**(4), 377–393 (2015)
11. Pahl, C., Xiong, H., Walshe, R.: A comparison of on-premise to cloud migration approaches. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 212–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40651-5_18
12. Jamshidi, P., Pahl, C., Chinenyeze, S., Liu, X.: Cloud migration patterns: a multi-cloud service architecture perspective. In: 10th International Workshop on Engineering Service Oriented Applications - WESOA 2014 (2014)

13. Jamshidi, P., Pahl, C., Mendonça, N.C.: Pattern-based multi-cloud architecture migration. Softw. Pract. Experience **47**(9), 1159–1184 (2017)
14. Sharma, B., Thulasiram, R., Thulasiraman, P., Grag, S.: Pricing cloud compute commodities: a novel financial economic model. In: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (2012)
15. Pal, R., Hui, P.: Economic Models for Cloud Service Market (Pricing and Capacity Planning). Telekom Innovation Laboratories (2015)
16. Son, J.: Automated Decision System for Efficient Resource Selection and Allocation in Inter-Clouds. The University of Melbourne, Department of Computing and Information System (2013)
17. Abhishek, V. Kash, I., Key, P.: Fixed and market pricing for cloud services. In: 7th Workshop on the Economics of Networks, Systems, and Computation (NetEcon) (2012)
18. Gilia, P., Sood, S.: Automatic selection and ranking of cloud providers using service level agreements. Int. J. Comput. Appl. **72**(11), 45–52 (2013)
19. Fang, D., Liu, X., Romdhani, I., Pahl, C.: An approach to unified cloud service access, manipulation and dynamic orchestration via semantic cloud service operation specification framework. J. Cloud Comput. **4**(1) (2015). Article no. 14
20. Wang, W., Zhang, P., Lan, T., Aggarwal, V.: Datacenter net profit optimization with deadline dependent pricing. In: 46th Annual Conference on Information Sciences and Systems (CISS) (2012)
21. Giardino, C., Bajwa, S.S., Wang, X., Abrahamsson, P.: Key challenges in early-stage software startups. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) XP 2015. LNBIP, vol. 212, pp. 52–63. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_5
22. Fang, D., Liu, X., Romdhani, I., Jamshidi, P., Pahl, C.: An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation. Future Gener. Comput. Syst. **56**, 11–26 (2016)
23. Jamshidi, P., Ghafari, M., Ahmad, A., Pahl, C.: A framework for classifying and comparing architecture-centric software evolution research. In: 17th European Conference on Software Maintenance and Reengineering (CSMR 2013), pp. 305–314. IEEE (2013)
24. Li, H., Zhong, L., Liu, J., Li, B., Xu, K.: Cost-effective partial migration of VoD services to content clouds. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 203–210. IEEE (2011)

A DMN-Based Approach for Dynamic Deployment Modelling of Cloud Applications

Frank Griesinger[1(✉)], Daniel Seybold[1], Jörg Domaschka[1], Kyriakos Kritikos[2], and Robert Woitsch[3]

[1] Institute of Information Resource Management, University of Ulm, Ulm, Germany
{frank.griesinger,daniel.seybold,joerg.domaschka}@uni-ulm.de
[2] ICS, FORTH, Heraklion, Greece
kritikos@ics.forth.gr
[3] BOC Asset Management, Vienna, Austria
robert.woitsch@boc-eu.com

Abstract. Cloud computing is well suited for applications with a distributed architecture and dynamic demand of resources. Yet, current approaches to model cloud application deployment do not cater for the application's dynamic nature and its rapidly changing business requirements. The static description of deployments results in a lack of reusability and also lacks an integrated way to adapt to the current context. To reuse and refine the deployment model, we introduce a *simple decision layer* on top of a cloud application description, which abstracts from the actual deployment language and allows assembling the deployment model from existing model fragments. Those fragments are chosen based on the input of the decision process. We define an architecture for the decision layer and sketch an implementation based on CAMEL, DMN, and ADOxx. The benefits of the decision layer are illustrated by two use cases. Our approach shifts the focus from a static to a dynamic and reusable modelling process, which also reduces the modeller's effort.

Keywords: DMN · DevOps · MDE · Cloud · Deployment modelling

1 Introduction

The cloud computing paradigm promises the unlimited offering of computational resources in a pay-as-you-go model. This helps organisations, especially SMEs, with unplannable or highly dynamic resource demands, to dynamically reserve IT resources as needed without having a huge upfront investment.

The benefits of cloud computing come along with an additional technical depth, which may hinder the migration to the cloud. Hence, industry and academia investigate approaches easing the cloud adoption. A well established approach to reduce technical complexity is model-driven engineering (MDE). Within MDE, domain specific languages (DSLs) for the cloud computing domain

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 104–111, 2018.
https://doi.org/10.1007/978-3-319-72125-5_8

evolved, including TOSCA [4] and CAMEL [5]. Such DSLs ease the cloud adoption by enabling a complex cloud application deployment model on a higher level. A cloud orchestration tool (COT), such as Cloudiator [2] can then process this deployment model.

Still, the specification of a deployment requires a certain degree of technical knowledge to create the deployment model, which is static in nature. Current modelling approaches do not reflect the dynamic in changing business requirements that impact an implemented deployment model at run-time. As shown in the first lane of Fig. 1, any requirement change leads to the remodelling of the deployment model, which is error-prone and cost-intensive.

In this position paper, we propose a novel approach by adding *(i)* dynamic and *(ii)* reusability to cloud application modelling. We introduce a simple decision layer on top of the modelling, enabling the transformation of business requirements into a technical deployment model at both design- and run-time as shown in the second lane of Fig. 1. Based on two use cases, we demonstrate how our approach enhances the scope of cloud application modelling and therefore eases cloud adoption.

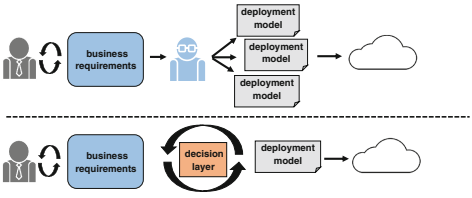


Fig. 1. Dynamic deployment modelling

The remainder of the paper is structured as follows: Sect. 2 analyses the problem. Section 3 presents our approach and sketches an implementation. The usage is presented on two use cases in Sect. 4. Section 5 discusses the approach, while Sect. 6 summarises the related work. Section 7 concludes the paper.

2 Problem Statement

Modelling cloud applications is technically challenging and therefore error-prone. In addition, most approaches have a steep learning curve. The process of current modelling approaches is a static sequence of the steps (cf. Fig. 1): *(i)* business experts define high level business requirements, *(ii)* technical experts manually map these requirements to a technical deployment model using a cloud DSL, *(iii)* the deployment model is put into a COT. The shortcomings of this process are founded in the dynamic nature of cloud applications, and as soon as a requirement changes, the complete process has to start from the beginning. Due to its complexity, the repetition of all steps is cost-intensive and error-prone. This is also caused by the fact that current cloud modelling approaches do not focus on the automated reuse of model fragments and manual involvement increases the risk of failure. In addition, decisions are only implicitly integrated and hard-coupled into the deployment model. This hinders the employment of a feedback loop in this process to re-evaluate the business requirements when needed.

The following scenarios exemplify the shortcoming of a static procedure: *(i)* A customer-specific deployment model that requires minor adjustments, concerning the cloud provider or data location, will result in an independent model

per customer. *(ii)* Update roll-outs are an important feature of DevOps tools. With the release of a version the deployment model changes. In cloud modelling approaches, there is no support to model the dependency of the version in respect to the deployment model. *(iii)* The service configuration, such as cloud provider or virtual machines specification, highly depends on business requirements, such as the available budget.

We argue that these shortcoming can be removed by adding *(i) dynamic* and *(ii) reusability* to the modelling process.

3 Dynamic Cloud Modelling

This sections is structured as follows. First, we present a solution for adding dynamic to the modelling process. Followed by a realisation sketch of this solution.

3.1 Introducing a Decision Layer

We propose a novel approach to ease the process of cloud application modelling by adding a *simple decision layer* on top of existing cloud DSLs. The proposed decision layer operates between higher level *(business) requirements* and low level concrete *model fragments*. Business requirements are integrated as influencing factors of a decision process which maps them to concrete model fragments. These model fragments are used to assemble the deployment model.

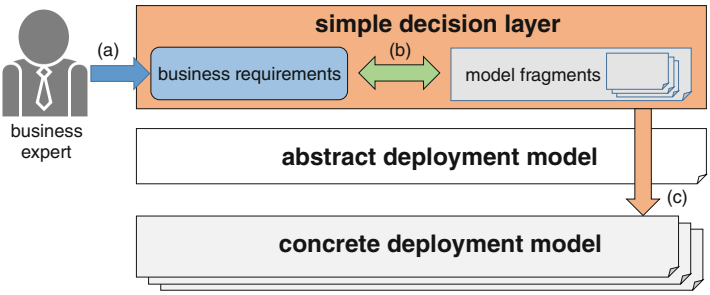


Fig. 2. Simple decision layer on top of deployment model.

As shown on Fig. 2, the simple decision layer handles business requirements and concrete model fragments. The decision layer abstracts from the actual language, by using an *abstract deployment model* that comprises anchor points for a decision process to concretise the deployment model by the output values of an evaluated decision. In contrast to the *concrete deployment model*, it is not completely described and therefore not executable in a COT. The business requirements are fed into the layer by business experts (Fig. 2(a)). The mapping

of requirements to model fragments is done based on a business knowledge model (Fig. 2(b)). When requirements arrive or change, the decision layer executes a decision process. After evaluating the business requirements, appropriate model fragments are selected and used to enhance the abstract deployment model and create a concrete deployment model that is executable by a COT (Fig. 2(c)).

Based on the requirements, the decision layer operates on an abstract deployment model and a *decision set* that is defined just once, to reuse existing model fragments in order to create multiple concrete deployment models.

While in current approaches the concrete deployment model has to be modelled per business requirement set, this approach requires to model one abstract deployment model and define its business knowledge model for arbitrary requirement sets. In order to extend available business requirements, additional mapping decision can be added to the business knowledge model. The simple decision layer is then able to reuse those model fragments for new incoming business requirements, as well as for other abstract deployment models.

In order to deal with changing requirements at run-time, the decision layer will reevaluate the decisions and update the respective model fragments. Thus, the proposed decision layer enables dynamic redefinition of the required model fragments based on the predefined decision set.

3.2 Realisation Sketch

We propose a realisation of our approach based on the Decision Model and Notation (DMN) [3] as decision layer and CAMEL [5] as the cloud modelling DSL.

The DMN standard provides a human-readable common notation for modelling and automating decisions. We choose decision tables (DTs) to represent decisions as these are well known to business experts. An example of a DT is shown in Table 1. A DT consists of three column types: *(i)* a hit policy, *(ii)* an input variable set, and *(iii)* an output variable set. The hit policy defines the selection over overlapping decisions with policies like **U**nique, i.e., only a single decision will be selected or **C**ollect, i.e., all decisions can be selected. Each input variable can potentially map to a respective output variable of a sub-decision table. Hence, there is a possible cascade of decisions leading to hierarchical decision tables. Any DT is associated with a business knowledge model (BKM) defining the decision logic, i.e., the mapping between the input and

Table 1. Image and Region DT

Hit Policy	Input		Output	
C	Privacy level string	Provider string	VM image string	Region string
1	Low	Provider X	Image X	US
2	Low	Provider Y	Image Y	Europe
...	...	...	...	...

output parameters. DMN is chosen as it is an impact gaining standard and it is already well adopted on the business level.

CAMEL models encompasses all technical details to deploy an application in the cloud, including specific cloud resources such as virtual machines and deployment structure. A concrete deployment model can be transformed into a set of cloud-provider specific deployment actions. We favoured CAMEL over other cloud application modelling languages like TOSCA as CAMEL supports the specification of a provider-independent deployment model, as well as an instance model.

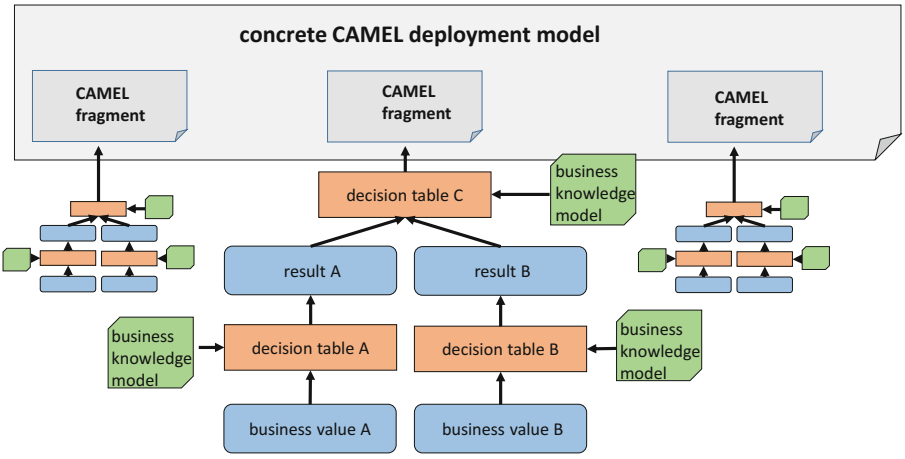


Fig. 3. Dynamic CAMEL modelling

Our proposed realisation is depicted in Fig. 3 implementing the decision layer as a hierarchical set of DTs enabling the dynamic CAMEL modelling. The DTs are specified by BKM fragments, which define the actual decisions in the DTs. We distinguish between two different types of output values, DMN results, used as input for other DTs, and derived CAMEL fragments.

Both languages are integrated into the meta-modelling platform ADOxx[1], providing a modelling tool for dynamically generating CAMEL models via DMN. ADOxx is able to provide a modelling user interface and the integration of algorithms to implement the usage of meta models. The main scenarios taken into consideration are *(i)* the specification of DMN via a graphical user interface, and *(ii)* the support of the execution of DMN to generate the CAMEL model.

4 Use Cases

We present use cases from the areas of DevOps and business-IT-alignment. We exemplify achieving their requirements by integrating our approach.

[1] https://www.adoxx.org/

4.1 Customer-Specific and Continuous Deployment

A common requirement in DevOps environments is having a *customer-specific deployment* that differs slightly due to customers' specific favors, and *continuous deployment* on version updates. Introducing a decision process enables in this case the reusability of the cloud application model. The application is only modelled once, but the concrete deployment model is generated dynamically for different requirements of the customers and of the application version.

A sample excerpt of a DT is shown in Table 1. The input of this DT are the trusted cloud provider and the privacy level. The output is the VM image and region for the model to be used for the service deployment.

4.2 Business Process as a Service

CloudSocket[2] introduces the concept of *Business Process as a Service (BPaaS)* by modelling business processes (BPs) on the highest level and semi-automatically align the BPs to the technical description of the required cloud services for the BP execution [8]. The BPaaS approach comprises a sequence of mappings from higher level business descriptions to low level technical descriptions. This chain contains points where decisions are made to create models of different levels of detail. The proposed layer caters for this mapping as it allows to integrate the business requirements to the model creation process. The decisions that have to be made in the BPaaS approach define, *(i)* which service to use, *(ii)* which configuration, such as cloud provider and hardware, the service will have and *(iii)* the service's behaviour at run-time.

5 Discussion

The presented approach has a major impact on evolving the current state of the art with respect to managing cloud applications through COTs and also for the features supported by COTs. As the input parameters for decision processes may change during run-time, COTs have to be able to update a deployed application on-the-fly according to the changes in the model. To implement this, a COT will need to create a change set between old model and new model and apply actions that implement the changes. This will involve adaptation actions currently not supported by any COT such as the migration of components onto different clouds.

In current modelling environments, the modeller directly interacts with a DSL or a direct (graphical) editor. Our approach shifts this view for modellers to a paradigm above the actual DSL. She will outline the deployment model by specifying the decisions that lead to the actual deployment. This will increase the reusability of cloud description fragments and cater for the dynamic nature of cloud-based applications and lower the learning curve for decision makers.

[2] https://www.cloudsocket.eu/

Although the paper motivates the importance of integrating business requirements in the decision process of cloud modelling, the presented approach is able to involve any kind of requirements, e.g. technical requirements, as the decision process is agnostic to the type of requirement. Also the rules described in the decision tables of the introduced layer can be translated into adaptation rules in the DSL. Obviously, this would demand the specification of the correlation between business requirements and e.g., the number of component instances in the case of scaling rules.

By applying our approach, application modeller can create abstract models and distribute them in a marketplace-like manner. Companies can choose from those abstract deployment models, customise them, and create concrete deployment models by the means of a company's specific business requirements. In contrast to similar application libraries of current COTs, our approach does not suffer from static models that needs low level adjustments to customise it.

6 Related Work

Besides DMN, there are numerous approaches for decision engines like Gandalf[3] or the IBM Operational Decision Management[4] that also apply decision tables to define business rules. Those can also be used to run the decision layer.

The usage of interconnected ordered decision tables as a selection method for cloud services can be categorised as a multi-criteria decision-making (MCDM) process [7]. In contrast to optimisation-based approaches realising MCDM decisions using utility functions, we are convinced that our approach is more user-friendly and -intuitive, due to the use of human-readable tables as interfaces.

Cloud orchestration tools mainly use DSLs to specify the deployment models [1]. However, they do not automatically create a set of differences to integrate modifications due to a decision process. This becomes necessary, when business requirements are evaluated on run-time and a feedback loop is integrated.

DevOps tools like Puppet[5] support updating an application at run-time on the basis of the differences between the latest and the currently active configuration. However, those DevOps tools operate on the level of the single component. They do not consider the overall view on the cloud-based application. The autonomous provisioning of infrastructure or platform resources is also out of scope of such tools. Our model-driven approach integrates the update functionality by using the application version as input for the decision process.

ToscaMart [6] introduces the idea of reusing model fragments for modelling by employing a marketplace of predefined application components to be used by the modellers to assemble their applications. This approach lacks a general decision-making process but instead relies on lowest technical requirements of the application to be assembled.

[3] https://gndf.io/
[4] http://www-03.ibm.com/software/products/de/odm
[5] https://puppet.com/

7 Summary

Cloud deployment models are described in domain specific languages (DSLs). Current DSLs are static and the creation comes along with the complexity of many technical details. Whereas modelling decisions are taken at design-time, influencing factors, such as technical or business requirements require to update fragments of the deployment model at run-time. Current modelling approaches only cater for updates of the complete deployment model and do not consider the reusability of constant model fragments.

In this position paper, we proposed a simple decision layer residing above current DSLs. This decision layer enhances the modelling scope by considering decisions affecting the deployment model at design and run-time.

We sketch a realisation based on the decision model and notation (DMN) as decision layer. DMN enables the semi-automatic creation of the deployment model from the results of DMN decision tables. Our realisation proposes the usage of DMN with the cloud DSL CAMEL in the ADOxx modelling environment. The suitability of the approach is discussed based on two use cases types.

Future work will encompass a prototype implementation of the presented decision layer based on the outlined technologies. Based on the prototype an evaluation on the impact of model creation and execution will be performed.

Acknowledgements. The research leading to these results has received funding from the EC's Framework Programme FP7/2007–2013 under grant agreement number 317715 (PaaSage) and the EC's Framework Programme HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

1. Baur, D., Seybold, D., Griesinger, F., Tsitsipas, A., Hauser, C.B., et al.: Cloud orchestration features: are tools fit for purpose? In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 95–101. IEEE (2015)
2. Domaschka, J., Baur, D., Seybold, D., Griesinger, F.: Cloudiator: a cross-cloud, multi-tenant deployment and runtime engine. In: 9th SummerSoC (2015)
3. Object Management Group: Decision model and notation. Technical rep., OMG (2015). http://www.omg.org/spec/DMN/1.1/
4. OASIS: Topology and Orchestration Specification for Cloud Applications Version 1.0 Committee Specification Draft 08 (2013)
5. Rossini, A.: Cloud Application Modelling and Execution Language (CAMEL) and the PaaSage workflow. In: Celesti, A., Leitner, P. (eds.) ESOCC 2015 Workshops. CCIS, vol. 567, pp. 437–439. Springer, Heidelberg (2016)
6. Soldani, J., Binz, T., Breitenbcher, U., Leymann, F., Brogi, A.: ToscaMart: a method for adapting and reusing cloud applications. J. Syst. Softw. **113**, 395–406 (2016)
7. Sun, L., Dong, H., Hussain, F.K., Hussain, O.K., Chang, E.: Cloud service selection: state-of-the-art and future research directions. J. Netw. Comput. Appl. **45**, 134–150 (2014)
8. Woitsch, R., Utz, W.: Business process as a service: model based business and it cloud alignment as a cloud offering. In: 2015 International Conference on Enterprise Systems (ES), pp. 121–130. IEEE (2015)

Cloud Migration Methodologies: Preliminary Findings

Mahdi Fahmideh(✉), Farhad Daneshgar, and Fethi Rabhi

University of New South Wales, Sydney, Australia
m.fahmidehgholami@unsw.edu.au, mehdi.fahmideh@gmail.com

Abstract. Research around cloud computing has largely been dedicated to addressing technical aspects associated with utilizing cloud services, surveying critical success factors for the cloud adoption, and opinions about its impact on IT functions. Nevertheless, the aspect of process models for the cloud migration has been slow in pace. Several methodologies have been proposed by both academia and industry for moving legacy applications to the cloud. This paper presents a criteria-based appraisal of such existing methodologies. The results of the analysis highlight the strengths and weaknesses of these methodologies and can be used by cloud service consumers for comparing and selecting the most appropriate ones that fit specific migration scenarios. The paper also suggests research opportunities to improve the status quo.

Keywords: Cloud migration · Legacy applications
Cloud migration methodology · Evaluation framework

1 Introduction

Cloud computing initiatives have received significant attention for addressing computational requirements of enterprise applications through offering a wide range of services which are universally accessible, acquirable and releasable on the fly, and payable based on the service usage. Many IT-based organizations are at the edge of moving their legacies to the cloud. While there are many valuable technical solutions to make legacies cloud-enabled, those solutions are not sufficient on their own and one should not undermine the equal importance of adopting a systematic methodology to enable legacies to benefits from cloud services. Such a methodology aids developers to organize the migration process and defines a step-by-step guidance on activities should be carried out to reengineer and move legacies to the cloud. This paper presents a review and evaluation of the extant cloud migration methodologies in the literature with the aim of understanding their features, strengths, weaknesses, and potential opportunities for future research. More than a dozen of cloud migration methodologies have been suggested by both from academia and industry. Some examples are Chauhan's Methodology [1], REMICS [2], Tran's Methodology [3], Cloud-RMM [4], Strauch's Methodology [5], Zhang's Methodology [6], Oracle [7], ARTIST [8], Amazon [9], and Legacy-to-Cloud Migration Horseshoe [10].

This paper is organized as follows: Sect. 2 develops an evaluation framework to assess the abovementioned methodologies, which is followed by Sect. 3 that reports

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 112–122, 2018.
https://doi.org/10.1007/978-3-319-72125-5_9

evaluation results of the methodologies. Section 4 discusses the implications and threats of this research. Finally, this paper concludes in Sect. 5.

2 Criteria-Based Evaluation of Migration Methodologies

In the context of software engineering, an evaluation framework constitutes a checklist of criteria (or methodological requirements) that an ideal methodology is expected to appropriately address them when dealing with a particular activity objective [11]. A methodology is checked against an evaluation framework in two steps: firstly, the methodology is scanned for features concerned by a criterion and then an evaluation result (e.g. scale point) is yield signifying the some extent that the methodology supports the criterion [11]. In order to ensure the quality of criteria set, the meta-criteria (criteria used to assess other criteria) proposed by Karam et al. [12] was taken into account to develop the proposed evaluation framework. Regarding to this source and context of this study, the criteria should be (i) sufficiently generic to cover a variety of scenarios for legacy application migration to the cloud regardless of a particular target cloud platform, (ii) distinct to characterise the similarities and differences of methodologies, and (iii) adequately comprehensive to cover end-to-end cloud migration process.

In developing of the criteria, we reviewed and synthesized various existing frameworks that define criteria attuned to evaluate software development methodologies in traditional software (re-engineering) engineering and software process improvement literatures. A set of criteria were identified in the studies by [12–16]. We also reviewed studies suggesting criteria pertinent to cloud migration methodologies. Studies by [3, 5, 17, 18] proposed essential criteria that an ideal cloud migration methodology should satisfy. This includes interoperability/portability, incompatibility resolution, cloud provider selection, and re-architecting, and tailorability. Once the criteria in the above sources were analyzed and redundancy and overlapping among them were removed, nineteen distinct criteria were shortlisted for the purpose of the current study. The criteria help to contrast and compare existing methodologies. They are listed in Table 1 and described in Sect. 3. We do not claim that the proposed evaluation framework is comprehensive, however, such a framework has not been proposed yet in literature and our study provides a good starting point in assessing and comparing extant cloud migration methodologies to highlight their strengths and weaknesses.

Table 1. Criteria expected to be supported by cloud migration methodologies

Criterion and Definition (letter C is the unique identifier of each criterion)		Type
Tailorability (C1): Providing mechanisms to configure and modify process or modelling language for a given project at hand.		Scale
Development Roles (C2): Defining roles who are responsible for performing migration activities or any stakeholder who are involved in a migration process.		Scale
Requirement Analysis (C3): Eliciting and specifying functionalities required to be fulfilled by cloud-enabled application such as computational, security, elasticity, and storage space requirements.		Scale
Legacy Understanding (C4): Recapturing an abstract As-Is representation of application architecture in terms of terms of functionality, different types of dependencies to other applications, interaction points and message follows between application components, as well as quality of code blocks for reuse and adaptation.		Scale
Cloud Service Selection (C5): Identifying, evaluating, and selecting a set of cloud providers that might suit organization and application requirements.		Scale
Re-Architecting	**Cloud Architecture Model Definition (C6**: Identifying components of legacy that are suitable for migration and defining their deployment in the cloud environments.	Scale
	Refactoring and Incompatibility Resolution (C7): Identifying and resolving incompatibilities between legacy components and cloud services.	Scale
	Enabling Application Elasticity (C8): Providing support for dynamic acquisition and release of cloud resources.	Scale
	Enabling Multi-Tenancy (C9): Providing support for enabling multi-tenancy in the application in terms of security, performance, customizability, and fault isolation, which might incur by running application in the cloud.	Scale
Deployment (C10): Adjusting the application and network configuration for the target cloud environment.		Scale
Monitoring (C11): Continuous monitoring of application and cloud resources to assure SLAs.		Scale
Test (C12): Defining activities for test and continuous delivery.		Scale
Work-Products and Notations (C13): Specifying work-products to be produced as outcomes of migration activities.		Scale
Modelling Language (C14): Specifying a modelling or notational component		Boolean
Unit of Migration (C15): Applicability of the methodology for the migrating different tier of a legacy application.		Multiple Answer
Migration Type (C16): Migration types are concerned with methodology.		Multiple Answer
Tool Support (C17): Availability of tools to support the methodology's activities and techniques.		Scale
Maturity (C18): Available account on successful adoption the methodology in real-world migration scenarios.		Multiple Answer

3 Analysis of Results

Table 2 summarizes the evaluation results of the methodologies according to the framework. The review of the methodologies reveals that many criteria; specifically Tailorability (C1), Development Roles (C2), Cloud Architecture Definition (C6), Refactoring (C7), and Multi-tenancy (C9) are not adequately supported. The methodologies do not comprehensively elaborate on activities related to these criteria, as a part of their mainstream process, which should be carried out to make a legacy application cloud-enabled. The following, delineates the results of our analysis and suggests areas that indicate future research directions to improve existing methodologies.

Table 2. Results of evaluating cloud migration methodologies

Criterion	Chauhan's Methodology	REMICS	Tran's Methodology	Cloud-RMM	Strauch's Methodology	Zhang's Methodology	Oracle	ARTIST	Amazon	Legacy-to-Cloud Migration Horseshoe
C1	○	◒	○	○	○	○	○	◒	○	○
C2	●	◒	○	○	○	○	○	●	○	○
C3	●	●	○	◒	◒	○	●	●	●	●
C4	◒	●	○	◒	◒	●	○	●	●	●
C5	●	○	●	◒	●	●	○	●	●	●
C6	●	◒	○	◒	○	○	○	○	◒	○
C7	◒	◒	◒	◒	●	○	●	◒	◒	○
C8	○	○	○	◒	○	○	○	○	●	○
C9	○	○	○	◒	○	○	○	○	○	○
C10	○	◒	●	◒	◒	●	●	●	●	○
C11	○	○	○	◒	○	○	●	●	●	○
C12	◒	●	●	◒	○	○	●	●	●	○
C13	●	●	○	○	○	◒	○	●	○	●
C14	N	Y	N	N	N	Y	N	Y	N	Y
C15	AS	AS	AS	NS	DL	AS	AS	AS	AS	AS
C16	II	II	IV	NS	IV	II	IV	II	V	NS
C17	○	●	○	○	◒	○	○	●	◒	○
C18	RE	CS	RE	NV	CS	CS	RE	CS	CS	NV

● *Fully Supported* explicitly supported by the method, ◒ *Partially Supported* by the method, ○ *Not Supported* by the method (neither a partial definition nor explanation for a requirement), N:No, Y:Yes, NV: Not Validated, CS: Case Study, RE: Reported Experience, AS: Application Stack, DL: Data Tier, NS: Not Specified, I, II, III, IV, and V: See definitions for the requirement Migration Type in Section 4.

Tailorability (C1). It has been well-acknowledged that in every methodology there are good features to adopt as well as deficiencies to avoid [14]. These features circumscribe the applicability of a methodology in a given project situation at hand. The fact that methodologies should be tailored or designed to suit the characteristics of a given cloud migration scenario is pinpointed in the cloud migration literature [18, 19]. As shown in Table 2, except for REMICS and ARTIST methodologies which provide a partial support for the tailorability, none of the existing methodologies offers mechanisms to fine-tune their processes or to check if the methodology is properly applicable for a migration scenario at hand. REMICS is structured in the form of a set of reusable method fragments which eases its tailoring through selecting suitable method fragments and assembling them with respect to a migration scenario. ARTIST offers a tool which

facilitates configuration and instantiation of the methodology for a given migration scenario. However, none of them provides explicit guidance on how to tailor or create a situational methodology out of the base methodology.

Development Roles (C2). While methodologies describe what activities are to carry out, the roles and required expertise that take these activities become a concern for its users. The definition of roles assists developers who have limited experience and are not sure about roles involving in a migration process. In spite of necessity of defining roles in any software development lifecycle, the majority of cloud migration methodologies do not specify roles involving during the migration process. As shown in Table 2, except for Chauhan's methodology and ARTIST, the definition of roles and their responsibilities have been neglected in the existing methodologies. In these methodologies, the definition of roles is borrowed from traditional software development and they do not define roles that might be cloud-specific.

Requirement Analysis (C3). Requirements specify the desired features that should be fulfilled by moving legacies to the cloud. Conventional requirement analysis techniques such as interview, prototyping, and workshop are widely used by REMICS, ARTIST, and Chauhan's methodology. Additionally, Oracle and Amazon extend the requirement analysis to focusing on computing requirements and application scalability. Furthermore, Legacy-to-Cloud Migration Horseshoe is concerned with inter-operability requirements of the target application between cloud platforms. Trans and Zhang's methodologies do not define any activities related to the requirement analysis.

Legacy Application Understanding (C4). This is common that the knowledge about legacy applications is undocumented and incomplete. An in-depth understanding of the current state of legacies helps to identify any characteristic that might influence the cloud migration process. Activities related to the legacy understanding are covered by most of the methodologies, except for Tran's and Oracle's methodologies. A few of reviewed methodologies such as Chauhan's methodology, Cloud-RMM, Strauch's methodology define activities related to recover of legacy architecture model but do not narrow to provide adequate mechanisms or guidelines to conduct them.

Cloud Platform/Service Selection (C5). Developer should not neglect the influences of selecting cloud platforms on the development effort and cost required for the migration process. A better compatibility between the legacy and cloud services can make the migration process very easy and shorter. For example, Tran's methodology reports a breakdown of activities for moving a .NET 3-tier application to run in Windows Azure. She highlights required development efforts for modifying data tier, code refactoring, and installation is major if the underlying technology of the legacy and cloud platform are not compatible with each other. Table 2 shows that all of the reviewed methodologies incorporate activities related to cloud service selection. However, REMICS and Oracle are at the other end of the spectrum: they do not provide any guidelines as to how cloud service can be evaluated and selected.

Re-architecting (C6, C7, C8, C9). Several important aspects should be incorporated into the migration process when re-engineering a legacy to a cloud platform. The first (C6) is to select suitable legacy components and define their new deployment model in the cloud on basis of concerns such as network latency, data transfer, data privacy, legal restrictions while satisfying the expected QoS of the whole application. Cloud migration methodologies can be examined with respect to their support for activities and guidelines to define a cloud architecture model of an application and move components to the different cloud servers. Only Chauhan's methodology defines this activity in its process model. The second architectural aspect (C7) is the resolving of incompatibilities that might occur between the legacy application and selected cloud platform/services. The incompatibilities might be sourced from mismatch between legacy codes and cloud service APIs, interface signatures, data types, and query calls. A methodology is expected to provide activities to identify possible incompatibility issues and accordingly proper guidelines to resolve them. Otherwise legacy will not be able to utilize cloud services. Back to Table 2, the criteria refactoring and incompatibility resolution is only supported by Strauch's methodology and Oracle methodology, though they focus on activities to resolve incompatibilities between the legacy database tier and a target cloud database service. Other methodologies suggested by Chauhan, REMICS, Tran, Cloud-RMM, ARTIST, and Amazon suffer from cursory definitions of code refactoring. The third architectural design aspect (C8) is to enable the legacy in a support of dynamic resource acquisition and release when it is running in the cloud. According to Table 2, activities related to the enabling elasticity in the legacies are only supported by Amazon methodology. The fourth aspect (C9) is the multi-tenancy support. A key concern in the re-architecting of legacy to address multi-tenancy is to provide a support in the application for isolating the security, performance, customizability, and fault of tenants. Without such a support, a migrated application may face the risk of tenant interference. As shown in Table 2, the majority of methodologies are silent regarding the multi-tenancy requirement. Cloud-RMM includes the multitenancy, however, it does not elaborate on how conducing it.

Deploying and monitoring (C10, C11). It is likely that the connection between the migrated legacy to the cloud and local network to be required. A methodology should properly define activities related to the network configuration such as the setting open ports, firewall policies, and connection strings to data and application (C10). The deployment is covered by all the methodologies except for Chauhan's methodology and Legacy-to-Cloud Migration Horseshoe. Besides, once migrated to the cloud, the continuous monitoring of the application and cloud resources to assure SLAs is necessary (C11). Only three methodologies Oracle, ARTIST, and, Amazon support this criterion.

Test (C12). Test is to ensure that the cloud-enabled application meets the goals of cloud migration. All the methodologies except for Strauch's methodology, Zhang's methodology, and Legacy-to-Cloud Migration Horseshoe define activities in coherence with the methodology to ensure that application conforms to the expectation of the cloud migration such as performance or resource utilization.

Work-products and modelling language (C13 and C14). An integral part of every methodology is to specify necessary work-products as the outcome of each activity throughout the process model. Defining work-product becomes important if automatic code generation is required for a specific cloud platform or users of methodology aim to trace or keep the list of work-products that have been produced throughout the migration process. With respect to this, among the reviewed methodologies, Chauhan's methodology and ARTIST explicitly define work-products as a result of performing each migration activity. However, REMICS, Zhang's methodology, and Legacy-to-Cloud Migration Horseshoe only defines representing the legacy architecture. Methodologies may specify modelling techniques along with a particular notation to represent the outcome of each development activity. Modelling techniques, however, is hardly supported in the existing methodologies. ARTIST and REMICS use UML for their whole lifecycles along with Zhang's methodology and Legacy-to-Cloud Migration Horseshoe that, respectively, use SoaML and Graph-based modelling to represent legacy architecture.

Unit of Migration (C15). Some organizations may not move the whole legacy stack to the cloud because of security concerns, rather they may migrate some legacy components to the cloud whilst other components are kept in local organization network and cloud services are offered to them. In this regard, it is important to investigate if a methodology is appropriate for moving a particular tier or whole legacy stack to the cloud. Given that, eight reviewed methodologies have been designed for full migration to the cloud. Strauch's methodology is particularly designed for moving legacy data tier to a cloud database solution.

Migration Type (C16). Regarding the common service delivery models IaaS, SaaS, and PaaS, one can view that there are several variants that a legacy can utilize cloud services. We defined the followings migration variants and assess if the methodologies support them. In *Type I* the business logic tier of a legacy (e.g. WS-BPEL), which offers discrete and reusable functionality, is deployed in the cloud infrastructure. In this migration type, the data tier is kept in local organization network. Deploying an image processing component of a legacy in E2C is an example of this migration type. In *Type II* some components or whole application stack is replaced with an available and fully tested cloud service. The Salesforce CRM application is a typical example of this type of cloud migration. In this review, Chauhan's methodology, REMICS, Zhang's methodology, and ARTIST support this type of migration. In *Type III* legacy database is deployed in a cloud data store provider. The components related to business logic tier are kept in local organization network and the database is deployed in public cloud data store such as Amazon Simple Storage Service, Amazon Elastic Block Store, Dropbox, or Zip Cloud. There is not methodology to support this migration type. In *Type IV* the data tier of a legacy is modified and converted to a cloud database solution such as Amazon SimpleDB, Google App Engine data store, or Google Cloud SQL. Tran's, Strauch's methodology, and Oracle support this migration type. Finally, in *Type V* the whole legacy stack is deployed in the cloud infrastructure where the legacy is encapsulated in a single virtual machine

and then run in the cloud infrastructure. From the reviewed methodologies, Amazon defines activities to carry out this migration type. Obviously, on the basis of a chosen migration type, different activities might be required to be carried out and accordingly a methodology should properly address them.

Tool support (C17). The adoption of a methodology is facilitated if it offers its own supportive tool or alternatively refers developers to existing third-party tools available in the cloud marketplace. Only ARTIST and REMICS provide tool for whole migration process model. More specifically, ARTIST proposes Eclipse-based suites which are integrated with its activities. Since produced work-products are stored in a shared repository, they can be accessed and modified by other tools. REMICS includes a set of tools that can be classified in the areas such as requirement management, knowledge recovery from legacies, re-transformation of legacy components to cloud architecture, and model-based testing. On the other hand, Strauch's and Amazon's methodologies offer tool for legacy architecture recovering, data migration, and resource elasticity management. Other methodologies do not offer any tools.

Maturity (C18). Validating a methodology in real-world migration scenarios and subsequently refining it through feedback from experts improves its applicability and maturity. In this review, the majority of methodologies, except for Cloud-RMM and Legacy-to-Cloud Migration Horseshoe, have reported the applying the methodology in a real-world example. An observed issue during the assessment was the lack of sufficient contextual information on the environment in which the methodology had been applied, description of techniques used to data collection and analysis, and addressing threats to validity.

4 Discussion

This section discusses research implications and possible threats to validity of the evaluation results.

Research implications. This research has two major contributions to the cloud migration literature. Firstly, the proposed evaluation framework serves as a valuable tool for project managers to assess and compare the capabilities of cloud migration methodologies and select ones which satisfies their migration scenario characteristics through reusing the strengths. They can also priorities the proposed requirements on the basis of their goals and evaluate methodologies with respect to these priorities. Secondly, the evaluation results can be used as a basis for the purpose of situational cloud migration methodology construction meaning that useful method fragments from the existing methodologies can be selected and assembled to create bespoke methodology that fits the characteristics of a migration scenario at hand.

Threats to evaluation validity. In spite of our effort to provide a comprehensive and objective comparison, some threats still exist as mentioned in the followings:

Conclusion validity. The evaluation results in this research are mainly theoretical and based on the available and published documents of the methodologies. However, a real evaluation of the methodologies through applying them in the same real-world migration scenario could yield to other results. However, such empirical assessment is planned as our future work. Furthermore, as the methodologies may have been yet improved by their designers, the evaluation may need to be updated.

Construct validity The validity of the evaluation results may be concerned in terms of measures that have been applied to assess the satisfaction of the methodological requirements. To minimize inconsistency in measuring, the definitions of criteria were used during assessment process (Table 1). These definitions checked the existence of activities, work-products,or roles that are related to the criteria.

Internal validity. A threat to the validity of this research is that the evaluation process was conducted by a single researcher. Hence, the evaluation results might be to some extent subjective in nature or undergone by misinterpretation. This threat can be further minimized if a Delphi technique [20] is applied where the evaluation process is performed by authors of the framework and subject matter experts. The difference between ratings can be resolved through a post-hoc evaluation discussion to reach a consensus.

External validity. We acknowledge that to assure generalizability of evaluation results, more evaluation with a higher number of domain experts is necessary. Furthermore, we selected a representative number of cloud migration methodologies that have been proposed in the literature. Nevertheless, more research on the evaluation of cloud migration methodologies and criteria which may have not been investigated in this research is required.

5 Future Work and Conclusion

This paper argued that the current state of legacies to the cloud needs to adopt methodological/process model perspective. Following an overview of existing methodologies related to legacy application to cloud migration, an evaluation of them regarding a set of important criteria was presented. The evaluation results were presented in a structured format and revealed that the methodologies suffer from the lack of tailorability, defining roles and work-products involved in the migration process, and incorporating a modelling language to model the output of activities. Additionally, there is no methodology which focuses on the migration types I and III. The cloud migration methodologies seem quite nascent and are yet to reach a high level of maturity. The current situation of the cloud migration methodologies definitely calls for further research aimed at ameliorating the status quo. With respect to the evaluation results in this paper, a further research opportunity is to develop a new cloud migration methodology through reusing the strengths of existing methodologies while addressing their deficiencies. This can be based on identifying method fragments from the methodologies and storing them in a method library. Such a harness can be effectively addressed by adopting the idea of

situational method engineering approach [21] where cloud migration solutions in the literature can be abstracted away and structured as a complementary source for developing method fragments. Once such reusable method fragments identified, they can be further assembled to construct customized migration methodology which fits a given migration scenario.

References

1. Chauhan, M.A., Babar, M.A.: Towards process support for migrating applications to cloud computing. In: 2012 International Conference on Cloud and Service Computing (CSC), pp. 80–87 (2012)
2. Mohagheghi, P.: Software engineering challenges for migration to the service cloud paradigm: ongoing work in the REMICS project. In: 2011 IEEE World Congress on Services (SERVICES), pp. 507–514 (2011)
3. Tran, V., Keung, J., Liu, A., Fekete, A.: Application migration to cloud: a taxonomy of critical factors. In: Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, pp. 22–28 (2011)
4. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **8**, 1 (2013)
5. Strauch, V.A.S., Karastoyanova, D., Leymann, F.: Migrating enterprise applications to the cloud: methodology and evaluation. Int. J. Big Data Intell. **5**, 127–140 (2014)
6. Zhang, W., Berre, A.J., Roman, D., Huru, H.A.: Migrating legacy applications to the service Cloud. In: 14th Conference companion on Object Oriented Programming Systems Languages and Applications (OOPSLA 2009), pp. 59–68 (2009)
7. Laszewski, T., Nauduri, P.: Migrating to the Cloud: Oracle Client/Server Modernization. Elsevier, New York (2011)
8. Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., Orue-Echevarria, L., Alonso, J., et al.: ARTIST methodology and framework: a novel approach for the migration of legacy software on the Cloud. In: 2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 424–431 (2013)
9. Varia, J.: Migrating your existing applications to the AWS cloud: a phase-driven approach to cloud migration (2010)
10. Ahmad, A., Babar, M.A.: A framework for architecture-driven migration of legacy systems to cloud-enabled software. Presented at the Proceedings of the WICSA 2014 Companion Volume, Sydney, Australia (2014)
11. Kitchenham, B., Linkman, S., Law, D.: DESMET: a methodology for evaluating software engineering methods and tools. Comput. Control Eng. J. **8**, 120–126 (1997)
12. Karam, G.M., Casselman, R.S.: A cataloging framework for software development methods. Computer **26**, 34–44 (1993)
13. Wood, B., Pethia, R., Gold, L.R., Firth, R.: A guide to the assessment of software development methods. DTIC Document (1988)
14. Ramsin, R., Paige, R.F.: Process-centered review of object oriented software development methodologies. ACM Comput. Surv. (CSUR) **40**, 3 (2008)
15. Sturm, A., Shehory, O.: A framework for evaluating agent-oriented methodologies. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS -2003. LNCS (LNAI), vol. 3030, pp. 94–109. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25943-5_7
16. Tran, Q.-N.N., Low, G.C.: Comparison of ten agent-oriented methodologies. In: Agent-Oriented Methodologies, pp. 341–367 (2005)

17. Quang Hieu, V., Asal, R.: Legacy application migration to the cloud: practicability and methodology. In: 2012 IEEE Eighth World Congress on Services (SERVICES), pp. 270–277 (2012)
18. Andrikopoulos, V., Binz, T., Leymann, F., Strauch, S.: How to adapt applications for the Cloud environment. Computing **95**, 493–535 (2013)
19. Mahmood, Z. (ed.): Cloud Computing Methods and Practical Approaches. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5107-4
20. Okoli, C., Pawlowski, S.D.: The Delphi method as a research tool: an example, design considerations and applications. Inf. Manag. **42**, 15–29 (2004)
21. Harmsen, A.F., Brinkkemper, J., Oei, J.H.: Situational method engineering for information system project approaches. Department of Computer Science, University of Twente (1994)

Workflow Skeletons: Improving Scientific Workflow Execution Through Service Migration

Tino Fleuren(✉)

Fraunhofer Institute for Industrial Mathematics,
Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
tino.fleuren@itwm.fraunhofer.de
http://www.fraunhofer.de

Abstract. Planning the execution of a long-running scientific workflow orchestrating a huge number of services in a cloud infrastructure is a hard thing to do, because anticipating the current infrastructure situation at a given point in time is far from easy.

This paper describes a means for adjusting "workflow skeletons"-based scientific applications to this dynamically changing situation by using service migration allowing for moving services to different hosts, even if they already started execution.

From a global workflow perspective, there may be several reasons for deciding at runtime to move services to other hosts than originally planned. For example, sometimes it is not easy to predict in which phase of the workflow big data will be produced. In such situations, moving services to the data's location instead of staging data to the services will save time and network bandwidth. The concept of "workflow skeleton" is enhanced by letting single tasks decide about starting a migration based on a predefined set of policies.

Keywords: Reconfiguration of scientific workflows
Service migration · Workflow skeletons · Workflow building blocks
Workflow execution

1 Introduction

In general, scientific workflows are experimental, i.e., workflows are halted during execution, reconfigured or rescheduled. Moreover, they are long-running and process large amounts of data. Therefore, workflows have to be able to adjust to environmental changes; mechanisms for optimal reconfiguration of scientific workflows are needed.

There are at least two possible means for reconfiguration in cloud environments. First, clouds allow for dynamic allocation of resources. If the current resource situation changes, it may be necessary, especially for long-running workflows, to reschedule resource allocation or to move virtual machines to other hosts.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 123–134, 2018.
https://doi.org/10.1007/978-3-319-72125-5_10

The second option is to dynamically relocate services through service migration. The size of intermediate data produced by workflow activities, can rarely be determined before the workflow is started. Being able to migrate services to servers hosting these data avoids unnecessary transfer of data, because the size of the service's binary is small in comparison. The decision for migration has to be made at run time.

Problem Statement. After re-evaluating the current situation of the cloud infrastructure at a given point in time, service migration is a means to improve the workflow execution. Three issues have to be tackled: how, what, and when to migrate.

The first challenge is how to conduct a service migration. Scientific workflows usually consist of activities that are implemented using diverse kinds of service frameworks and technologies. For migrating a service, the systems must support remote deployment that may even be in another administrative domain. Thus, security issues like authorization and authentication must be taken into account.

The second challenge is to determine what data has to be copied to the new location. First of all, the service's binary must be copied to a new host. Next, if the services are already executing, they must be halted; all necessary data like input data, intermediate data, and session data has to be transferred, and then the service can continue its work at the interruption point.

The third challenge is when to migrate, i.e., how to find the best point in time for a migration. First rules and criteria need to be defined that help to decide whether and how to conduct a reconfiguration. Based on those rules, with the optimization goal in mind, the system must ensure to avoid unnecessary migrations, to not trigger migrations too often, and to not move services several time back and forth.

Contribution. In our former work we presented the concept of "workflow skeletons" [3]. The main contribution of this paper is the introduction of service migration to workflow skeletons by utilizing a policy-based decision system. The decision system is able to migrate services to the best suitable hosts while the workflow and even the services are executing. Our prototypical implementation ProWorkE that efficiently executes workflow skeletons on cloud resources has been enhanced for demonstrating the power of service migration for an example image rendering workflow.

2 Related Work

Several publications directly address the topic of service migration. In [11], the authors describe an approach for making Web services migratable. The framework allows for packaging the service and the associated data in a way that it can be restored at new locations. However, the migration decision – who is deciding when to migrate – is not discussed.

The authors of [12] present the use of self adaptive mobile processes (SAM-Proc) as an abstraction to ease application development for ubiquitous systems. Those processes are capable of changing their behavior and location over time

as specified in their own descriptive language. The application of the SAM-Proc framework can be implemented either using CORBA or using Web services both of which support weak migration. In [13], the same authors proposed the concept of mobile processes by means of self-adaptive migratable web services (SAM-WS).

In [10], the authors talk about making the web services mobile in Grid infrastructures. The authors also discuss the problem of tackling the dynamically changing platform and computing infrastructure for stateful services. Their solution allows stateful services to migrate while ensuring that already executed instructions need not be re-executed after migration.

In [7], a framework for service migration in cloud environments is presented using genetic algorithms for determining possible migration settings. Their algorithm utilizes a cost model with various service migration costs, including the costs of migration overhead. Like in our solution, migration information will be stored including locations of all migrated and replicated services. In addition, hosted services will be logged in a service registry.

The authors of [9] use the Semantic Application Design Language (SADL) for defining ontologies that describe, which services will be migrated using their migration framework (see [8]) and which servers could be used as migration targets. However, the solution relies on a particular migration decision strategy component that must be provided by a user utilizing the presented framework in contrast to our policy-based approach.

3 Background: Workflow Skeletons

In [3,4], we present a formal model of the concept "workflow skeletons" for specifying parallel parts of scientific workflows as building blocks. The concept of workflow skeletons is a refinement of the idea of algorithmic skeletons elevated from programming libraries to the higher level of workflows [1]. Workflow skeletons improve modeling, configuration, and execution of scientific workflows.

Workflow skeletons can be used on top of existing S-WfMS[1] – introducing a new abstraction in the specification without having to add new software entities to the system. Using workflow skeleton, the workflow is described on a higher abstraction level comprising skeletons as building blocks. Those descriptions can then be translated to the workflow language of the concrete target S-WfMS, i.e., for the actual workflow run this abstraction will be removed. This allows for a non-intrusive enhancement of the descriptive capabilities of workflow languages.

A workflow skeleton $\boldsymbol{S} = \langle P, C, i, o \rangle$ is a *directed acyclic graph (DAG)* comprising a set P of nodes called "proxies" and a set C of edges called "channels". Proxies are placeholders for the actual workflow activities. They receive data on input ports defined by the tuple i and present the data of its associated workflow activity at output ports defined by the tuple o. Channels are dataflow connections between proxies, i.e., links from one output port of a proxy to one input port of its successor proxy.

[1] Scientific workflow management systems.

Configuration of a workflow skeleton consists of several aspects: configuring the workflow activities, assigning them to cloud resources, and make use of data and/or task parallelism. Workflow skeletons are configured by assigning key/-value pairs to all the attributes of its proxies and channels, e.g. proxies need all information to invoke its workflow activities.

Workflow skeletons can be parametrized, i.e., the number of parts that build a skeleton – like proxies, channels, or sub-skeletons – can be adjusted dynamically. Parameters allow to adjust the *length* and/or the *width* allowing for a higher level of *scalability* [3]. In addition to the skeleton definition, its configuration can also be described using parameters. This gives more control for optimizing the use of resources reducing the complexity. For example, the skeleton can be configured such that the number of data transfers is reduced.

Based on a formal model of the concept of workflow skeleton we define a specification language for describing workflow skeletons: *Skeleton Workflow Language (WorkSKEL)* [4].

4 Service Migration

There are many definitions for "service migration"; in this paper, service migration is defined as follows:

Definition 1. *Moving a service, i.e. its code, session state (including internal and client state) as well as input and intermediate data from one server (host A) to another (host B) without causing service outages and loss of processed data.*

We use the terms *service*, *service invocation* and *workflow activity* as synonyms in the sense that they represent the steps of a workflow and are in nature distributed software components that can be invoked remotely. Services offer standardized interfaces to activities like invoking Web and Grid services, querying databases, starting scripts or legacy applications, submitting jobs to clusters and Grids, and others. In the context of scientific workflows, workflow activities are typically long-running and computationally intensive requiring high-performance resources. For workflow skeletons, an activity is represented by a so-called *proxy* - we say that a proxy is associated with a workflow activity (cf. Sect. 3).

The service migration may even take place across different networks and administrative domains as well as between heterogeneous environments. Thus, service migration is more challenging than just transferring the service's binaries from one location to another.

There are numerous reasons for introducing service migration to scientific workflow systems. For example, one reason might be to improve the performance of compute-intensive services by allocating more cloud nodes, if the load on currently used nodes is too high, and move parts of running service instances to new nodes, thus accelerating the overall scientific workflow.

The decision when and where to migrate services is difficult because multiple criteria come into play. The following enumeration includes some criteria that

may be a single factor or part of an arbitrary combination of any number of criteria. Here is a (not comprehensive) list of possible benefits for conducting a service migration:

- Performance improvements:
  - Parallelism: increase number of cloud nodes and distribute services evenly
  - Current load of the hosts (dynamic or static compensation)
- Fault tolerance for long-running workflows: e.g. consider maintenance schedules for servers connected to specialized hardware
- Availability of services: add new services to other compute nodes
- Reduction of large data transfers: cost of transferring the service to the data vs. the other way around
- Reduction of communication overhead between services: e.g. by grouping services that communicate a lot with each other on a common host
- Access to resources
  - Resources that are not accessible over a public network: e.g. services will have to use sensor data produced by specialized hardware that can only be reached via dedicated computers
  - Licensed software: the service needs access to licensed software that are secured with hardware dongles
  - Hardware resources: the service needs a computer with more memory, CPU power, etc.
- Costs: reduce costs by moving services to alternative resources. For example, cloud systems like AMAZON EC2 allow user to bid on spare Amazon EC2 computing capacity[2]. Or certain tariff system (night, day) will be applied
- Energy efficiency: computers will be shut down at nights.

5 Workflow Skeletons Supporting Service Migration

In this section, we first present our approach to answering the three question defined in the introduction: how, what, and when to migrate. Then we describe our service migration system based on condition/action policies.

5.1 How, What, and When to Migrate

How to Migrate. How to carry out a service migration depends on several aspects. One factor is the heterogeneity of the execution environment. In a hybrid environment consisting of clouds, local computers and campus/company networks, all kinds of software and hardware are installed and may be used as part of the scientific workflow. Therefore, not all of the compute nodes are suitable for a specific service technology, and not all S-WMSs or service engines support remote deployment. Additionally, when moving services to other nodes, administrative rights are required on the target node. From the target's point of view a migration is in general a deployment. This can be particularly problematic,

[2] Amazon EC2 Spot Instances, https://aws.amazon.com/ec2/spot/.

if the migration should be made possible between different administered systems, which is often the case collaborating scientists from different universities. Therefore, the system should be able to allow for building virtual organizations[3]. In such scenarios, the S-WMS must have the rights to deploy services and files to the target system, and of course access to the target's network.

In our solution workflow skeleton control services by proxies that can be easily configured with all necessary information for triggering and migrating its associated service; this includes credentials, certificates, etc. Proxies can be enhanced by plugins, so if additional behavior for a specific service technology is needed, a new plugin will be inserted. In our implementation the skeleton execution engine calls the migration manager, which is itself a special proxy that can be used to conduct migrations. It can be enhanced by adding additional plugins if need be.

What to Migrate. On the one hand, services can be *stateful or stateless*. Typically in service oriented architectures, stateless services are recommended [2]. However, in Grid environments services have to handle state, i.e., session state (also called client state). For example, the Web Services Resource Framework (WSRF) is a set of specifications that define the modeling and management of stateful resources using existing Web services technologies[4]. On the other hand, services can be *inactive or active*. Inactive services have no running instances, no state data, and can easily be migrated. For migrating an active service, the service's internal state has to be taken into account. When an instance is stopped, the internal state is saved and all data is transferred.

All combination of active/inactive, stateless/stateful, and internal/client state data will influence the **degree of difficulty** for conducting a service migration (see Fig. 1).

In all situations, at least the service binaries must be moved to the new location. Then the service will have to be deployed on the new host. This will require a mechanism for remote deployment of services giving rise to security challenges like authorization and authentication.

To ensure maximum flexibility, session state data and the current internal state, must be taken into account. The session data can be managed either by the client or the service itself. Migrating active services with internal or client state data, represents the greatest challenge. The service implementation may be multi-threaded, storing state in memory and/or databases as well as files. The simplest case of handling state is to wait for the end of all active instances of the service and perform the migration after completion.

[3] For example, Grid environments allow for defining groups, which may be distributed around the world, the so-called virtual organization (VO). Only members of the VO may access its resources. The Grid middleware's security mechanisms ensure the protection of resources and data.

[4] Web Services Resource Framework (WSRF), http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.

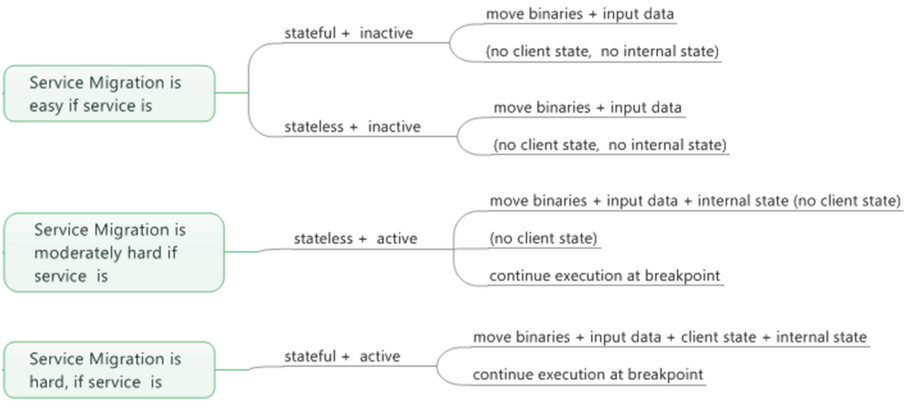


Fig. 1. Degree of difficulty

However, in our solution the proxies of workflow skeletons can control the migration of already running services for special services that are migratable. To this end, we provide a framework for making services **migratable**. A service developer uses the framework's API for making checkpoints, at which the execution can be interrupted and resumed. A checkpoint can store the service's binaries, session data, as well as input data and intermediate data. A function call is used to create a checkpoint at selected points in the service's logic. The developer has to ensure that all relevant state will be stored in the checkpoint data. After migration the framework uses the checkpoint data to resume the service's work at the exact point in logic, where it was interrupted. While the migration takes place, the proxies can reject incoming requests, or store them in a waiting queue. After the migration the proxy relays new request to the new service's location.

When to Trigger a Service Migration. An important factor in the migration of services is deciding when and where a service is to be moved. For workflow skeletons, such decisions can be made either manually or automatically. In certain cases, a manually triggering the service migration might be quite sensible. For example, when maintenance on the hardware or software system pending and running service instances would be affected, all services can be moved to non-affected node.

Workflow skeletons with automatic service migration achieve far more flexibility. The current condition of the dynamic execution environment needs to be considered. Network utilization, CPU load, memory size and other factors fluctuate sometimes fast and strongly; resources come and go freely. Usually, other software is using the resources at the same time. Adjustment to load and resource consumption are usually not coordinated and may therefore be in conflict.

In addition, it is important that a system supporting service migration will not trigger migrations too early or too often, because the migration will

have an overhead. Therefore, a flexible decision system is integrated based on condition-action policies (CA policies) that evaluates the current infrastructure situation and decides whether a migration should take place or not. For example, it makes little sense to frequently move services from one host to another if the migrations take longer than letting the service finish its work in a slower environment.

Therefore, we must clearly define the factors for automatic migration and calibrate its thresholds when to start migrations in the CA policies. Depending on the criterion either fixed thresholds or dynamically generated thresholds may be specified. In the latter case, for example, the system could consider the current network utilization and determine a suitable value for a threshold.

5.2 Policy-Based Decision Support

In our approach we use condition-action policies (CA policies) to define the behavior of the service migration system. This section gives an introduction to our CA policy documents.

Policy-Based Decision Support. For appropriate migration decisions, knowledge about the environment must be collected by monitors often for a longer period[5]. This data will be used together with pre-defined metrics in order to identify and estimate the potential for an workflow performance optimization gained by migrating services.

This evaluation is the task of the *decision algorithm* that will be loaded to the proxies as a plugin. The decision algorithm is configured by condition-action (CA) based policy documents. For example, for the criteria "processor load" the migration could be triggered if the CPU utilization $> 70\%$. The corresponding action could be to migrate a service to a pre-defined set of alternative hosts or to currently available hosts determined by the monitors data – or to allocate new cloud nodes.

Even with the capability of defining the migration behavior with policies, it will be a challenging task to specify suitable policies. Service migration should not be triggered too often. Therefore, for each policies thresholds are defined, which are variable and can be adjusted while the workflow is executing. After the service migration, this threshold can be raised to a higher value in order to avoid triggering another migration. It is also possible to define exclusions specifying that certain services cannot be migrated. For example, for a service with a short execution time (less than 1 min) the migration overhead may be too high. If several criteria are used for the migration decision, priorities must be defined and interactions between the individual criteria need to be taken into account. In some cases, an additional metric may be necessary to decide about an combination of several criteria. The overall goal of the service migration will result in

[5] Monitoring systems are out of scope of this paper.

different set of policies and thus in different behavior of the migration. For example, for the goal "minimizing energy consumption" the criterion *processing time* would play a minor role.

Policy Documents. The policy document is an XML file that acts as the input for the proxy's decision manager module. The configuration of the workflow skeleton will be enhanced by entries referencing the policy documents. With WorkSKEL, this can be done for single workflow activities or several at once [5]. When instantiating the workflow skeleton, the skeleton engine will be configured with policy documents.

Listing 1.1 shows an example policy document for server maintenance. Every day at 9:00 pm (starting from 2016/05/30) all instances of *RendererServiceOnServer1* will be migrated to the specified destination host. The *Condition* part defines rules, when the corresponding action should be triggered. The condition parameters will be used for specifying date and time of the migration execution, or for specifying the infrastructure parameters like percentage of CPU consumption, or temperature, or others that will trigger the action. Additionally, the condition's interval can be used to define repeating CA policies.

Policy condition parameters consist of a value (like DateTime, Integer, Float) and a value type (like percentage, Kbps, celsius) that explains how to interpret the value. In addition, the arithmetic comparator (like equals, greater than, less than) is specified that is used for comparing values. A weight can be defined for prioritizing CA policies that hold true simultaneously, thus, providing a means for finding a trade-off for multiple criteria.

```
<tns:PolicyConditionAction>
   <tns:PreDefinedConditionAction>
      <tns:Condition>
         <tns:Repeat>
            <tns:RepeatValue>1</tns:RepeatValue>
            <tns:RepeatUnit>Day</tns:RepeatUnit>
         </tns:Repeat>
         <tns:ConditionParameters>
            <tns:Parameters>
               <tns:Name>StartTime</tns:Name>
               <tns:Value>2016-05-30T09:00:00</tns:Value>
               <tns:ValueType>DateTime</tns:ValueType>
               <tns:ParameterComparator>Equal</tns:ParameterComparator>
               <tns:Weight>10</tns:Weight>
            </tns:Parameters>
         </tns:ConditionParameters>
      </tns:Condition>
      <tns:Action>
         <tns:ServerAndServicePair>
            <tns:DestinationIP>$$$$</tns:DestinationIP>
            <tns:ServiceLogicalName>RendererServiceOnServer1</
                tns:ServiceLogicalName>
         </tns:ServerAndServicePair>
         <tns:FailureNotification>true</tns:FailureNotification>
      </tns:Action>
   </tns:PreDefinedConditionAction>
```

Listing 1.1. Example Policy: Maintenance Policy

Currently, there are two variants of policies: pre-defined or monitor-defined. Pre-defined policies (as in Listing 1.1) cause the migration of services to specific, pre-defined hosts, whereas monitor-defined policies cause the algorithm to analyze the current server situation and migrate services to best suitable hosts. Search criteria for selecting servers that satisfy the specified conditions can be stated. One search criterion is used for finding source servers currently hosting services and the other one for finding destination servers as suitable targets for a migration. For example, the algorithm can search for servers that have a CPU utilization less than 60% and more than 70% free disk storage and use them as targets for migration.

6 Prototypical Implementation

The tool suite ProWorkE (Proxy-enhanced Workflow Engine) is a research prototype based on workflow skeleton that interprets WorkSKEL scripts and instantiates workflows that are composed of workflow skeletons [3,4].

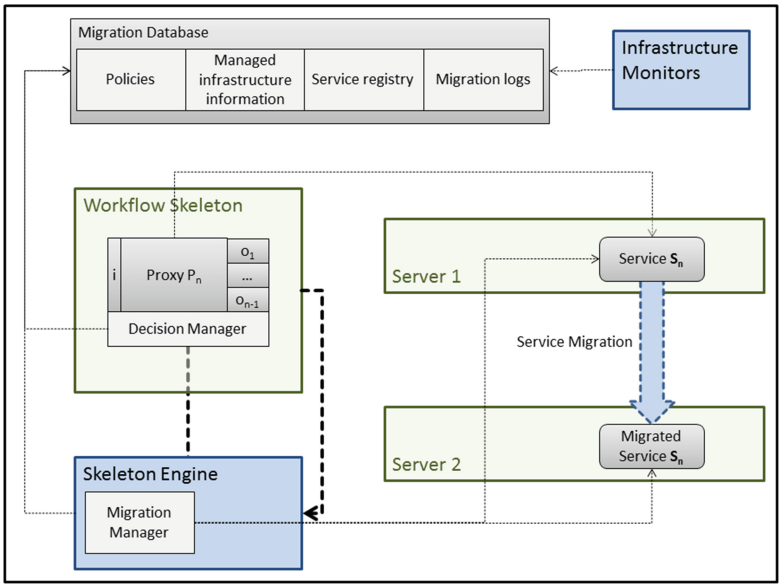


Fig. 2. System architecture of ProWorkE (new components enabling migration)

Figure 2 shows the architecture of the new components enabling service migration. The system will communicate with the computing infrastructure getting information about the current condition of the execution environment. The workflow skeleton and the proxies will be configured with the migration CA policies. All information about servers and services as well as logs about previous migrations will be kept in the migration database.

The decision manager is a plugin of ProWorkE's proxy services allowing for each proxy service to decide whether to migrate its associated service at run time. The proxies will make the decision to migrate based on the current environmental state given by the infrastructure monitors and the conditions defined in the policies. The migration manager is a new part of ProWorkE's "Skeleton Engine" that is implemented as a service itself and will be used by proxy services for executing a migration. WorkSKEL scripts allow to configure workflow skeletons with all necessary data like credentials and certificates that are needed to deploy services to foreign servers.

An image rendering workflow dividing a picture in tiles that can be rendered in parallel (cf. [6]) has been refactored with migratable versions of the rendering service. The workflow was deployed in a setting including servers from the AMAZON cloud[6] and private servers. For the private servers, we could connect our infrastructure monitors to gather information about the current environmental conditions – and the AMAZON nodes where used as destination servers.

Table 1. Comparison of the average service execution time (in milliseconds) of migratable and non-migratable service for 1 MB, 5 MB, and 10 MB data packets

Scenarios	1 MB	5 MB	10 MB	Average
Migratable	13519,26	13769,66	20825,70	16038,21
Non-Migratable	13448,10	16267,93	14147,36	14621,13

Several experiments with three scenarios have been conducted: non-migratable workflow execution as well as migratable workflow execution with and without service interruption. Each scenario has been started 30 times for three different input data sizes of 1 MB, 5 MB, and 10 MB. Table 1 shows the comparison of the average execution time (in ms) of the migratable and non-migratable render services under different input data loads. The execution times depend on the input size and the time the proxies need for un-marshaling the data and the fluctuations of the performance of the cloud infrastructure. However, the increase of execution time caused by migration overhead seems to be negligible for long-running workflow skeletons.

7 Conclusions and Future Work

In this paper, we showed an enhancement of workflow skeletons offering the possibility of reconfiguration by using service migration. We described the challenges of service migration especially for coming to a decision whether to migrate or not. An approach for decisions based on policy documents was presented taking the overall goal of the migration into account. All workflows finish their work

[6] see AMAZON Web Services (AWS): http://aws.amazon.com.

without noticing that some of its services have been migrated and that all service requests have been rerouted to new locations – even services that were already executing when receiving the migration request. However, defining CA policies well is hard; therefore, in next research steps we would like the system to detect conflicting policies and to learn from previous migration decisions and to use this information for adjusting the policies.

References

1. Cole, M.: Algorithmic Skeletons: Structured Management of Parallel Computation. MIT Press and Pitman, London (1989)
2. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)
3. Fleuren, T.: Workflow-Skelette: Konzeptionen zur Modellierung und effizienten Ausführung wissenschaftlicher Workflows. Verlag Dr. Hut (2014)
4. Fleuren, T., Götze, J., Müller, P.: Workflow skeletons: increasing scalability of scientific workflows by combining orchestration and choreography. In: 9th IEEE European Conference on Web Services (ECOWS) (2011)
5. Fleuren, T., Götze, J., Müller, P.: Facilitating scientific workflow configuration with parameterized workflow skeletons. In: Proceedings of 39th Euromicro SEAA, Santander, Spain. IEEE (2013). http://www.euromicro.org
6. Fleuren, T., Götze, J., Müller, P.: Workflow skeletons: a non-intrusive approach for facilitating scientific workflow modeling. In: Proceedings of 40th Euromicro SEAA, Verona, Italy. IEEE (2014). http://www.euromicro.org
7. Hao, W., Yen, I.L., Thuraisingham, B.: Dynamic service and data migration in the clouds. In: 2009 33rd Annual IEEE International Computer Software and Applications Conference, vol. 2, pp. 134–139, July 2009
8. Kazzaz, M.M., Rychlý, M.: A web service migration framework. In: The Eighth International Conference on Internet and Web Applications and Services, ICIW 2013, pp. 58–62. The International Academy, Research and Industry Association (2013)
9. Kazzaz, M.M., Rychlý, M.: Web service migration with migration decisions based on ontology reasoning. In: Proceedings of the Twelfth International Conference on Informatics - Informatics 2013, pp. 186–191. Faculty of Electrical Engineering and Informatics, University of Technology Košice (2013)
10. Marzouk, S., Jmaïel, M.: Towards making WSRF based web services strongly mobile. In: IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2008, pp. 192–197. IEEE (2008)
11. Meehean, J., Livny, M.: A service migration case study: migrating the condor schedd. In: Midwest Instruction and Computing Symposium (2005)
12. Schmidt, H., Hauck, F.J.: SAMProc: middleware for self-adaptive mobile processes in heterogeneous ubiquitous environments. In: Proceedings of the 4th on Middleware Doctoral Symposium, MDS 2007, pp. 11:1–11:6. ACM, New York (2007). Article no:11, ISBN: 978-1-59593-933-3. https://doi.org/10.1145/1377934.1377935
13. Schmidt, H., Kapitza, R., Hauck, F.J., Reiser, H.P.: Adaptive web service migration. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 182–195. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68642-2_15

Consumer-Driven API Testing with Performance Contracts

Johannes Stähli[1], Sebastian Lang[2], Fabian Kajzar[2], and Christian Zirpins[1(✉)]

[1] Faculty of Computer Science and Business Information Systems (IWI), Karlsruhe University of Applied Sciences, Moltkestr. 30, 76131 Karlsruhe, Germany
{stjo1031,Christian.Zirpins}@hs-karlsruhe.de
[2] SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
{Sebastian.Lang,Fabian.Kajzar}@sap.com

Abstract. Modern software applications are often based on a modular structure where services expose functions and data via an API. In an enterprise context, such APIs may be reused in varying contexts with alternative frontends and on different platforms. E.g., an intranet application may be reused by another department or as part of a public portal thereby migrating between different private clouds.

When migrating services, it is imperative to assure their qualitative characteristics. Expectations of application users need to be satisfied despite of changes in service usage context and provisioning platform. The problem is (a) to adequately verify qualitative expectations after migration and (b) to optimize service provisioning respectively.

In this position paper we discuss an adaptive API testing approach for reusable application services and APIs. Our work builds on a case study of application service reuse and migration at SAP SE. Subsequently, we propose *performance contracts* as a means to capture non-functional application requirements on user level. We utilize these contracts for *consumer-driven API testing* in order to verify and optimize the migration of services to different application contexts.

Keywords: RESTful web services and APIs
Non-functional consumer-driven contract testing · Application reuse
Cloud migration

1 Introduction

After adopting service-oriented architecture principles and cloud-based provisioning models, the application landscape of modern enterprises is turning into an ecosystem of application service interfaces (aka APIs). Such APIs are dynamically reused in a variety of application contexts to enable alternative user interfaces and devices or to address totally distinct user communities.

E.g., an instant messaging application that was initially intended for use by a single department evolves towards worldwide roll-out to all employees or even

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 135–143, 2018.
https://doi.org/10.1007/978-3-319-72125-5_11

makes its public debut as part of the enterprise web portal. Generally, in the course of evolution, characteristics and expectations of API clients are naturally evolving with respect to API usage. Some cases like going public also require application service migration to different cloud platforms.

As a consequence, changing application contexts might negatively affect the quality (e.g., performance, reliability, security) of application services and result in a subsequent failure to satisfy consumer expectations. This leads to the question, how application providers might anticipate the effects of changing application contexts beforehand in order to adjust service provisioning.

To this end, *consumer-driven contract testing* has been widely discussed as a pattern to foster mutual awareness and agreement of expectations and obligations during service evolution [9]. We propose to extend consumer contracts, a client perspective onto its concrete service usage behavior, with performance metrics. In turn, we enable providers to use the aggregated set of consumer performance contracts for automated performance tests against the current or future cloud deployment of the service implementation. Our results reveal a promising pattern to support enterprise API management.

The rest of this paper is structured as follows: Sect. 2 introduces a case study illustrating the challenges of application service reuse and migration. Section 3 gives a short overview of consumer-driven contract testing. Section 4 introduces our approach of *API performance contracting* to support reuse and migration within API ecosystems. Section 5 surveys related work. Finally, Sect. 6 summarizes the paper and gives an outlook on our future work.

2 Case Study

Recently, SAP is pushing its products into the cloud to provide a modern provision model. One of these products is HCP[1], a PaaS-offering providing in-memory database and application services to rapidly develop new services or extend existing ones. Beyond public offerings SAPs IT organization also uses SAP HCP for internal solutions. These include simple services modeling business transactions but also more complex scenarios such as instant messaging for sharing business-related content in realtime.

Most of these solutions follow a similar architecture. *Provider services* expose business data and functions as HTTP-based REST API [3]. These services run in a Java Runtime provided by SAP HCP.

Multiple *consumer applications* utilize these services by connecting to REST APIs. SAP provides its own UI Development Toolkit for HTML5 (SAPUI5) that is often used to develop web-based applications. However, provider services are also consumed by other services, desktop applications or applications using the native mobile development kit of Android, iOS or Windows. Further, a consumer application is not strictly limited to use just one service. It may also connect to multiple services composing their business functions.

[1] SAP HANA Cloud Platform, https://hcp.sap.com/.

Regardless of how many services are consumed and which implementation technology is used, the stability of all provider services has to be guaranteed. Within SAP IT, given functional requirements of provider services are ensured using open source testing tools. However, when it comes to non-functional requirements such as service performance that are more sensitive to the evolution of the application landscape, an adequate testing facility is yet missing.

SAP HCP offers four different compute unit sizes, each varying in the amount of CPU cores and memory, to adjust the performance of provider services. If a service is introduced to an application context for the first time, only few information about its minimal hardware requirements are known. The service is often started with the minimal available compute unit size. Performance limitations are revealed in production system when the number of parallel consumers exceeds a critical point. As a result the compute unit size is increased to the next size. This approach often leads to a bad user experience and may even result in a temporary downtime of the provider service, affecting all consumers.

Having proper performance tests is even more important when it comes to extending the functionality of an existing consumer, new consumers are added or the application has to be migrated to another environment. E.g., the *SAP Relay* application allows employees to exchange business critical information in realtime. Currently, SAPUI5 and iOS clients have been implemented, but other client applications will follow soon. It must be ensured that the roll-out of new clients will not affect the performance of existing applications using this service and that the performance expectations of new clients are also met. In addition, it is planned to migrate the provider application also to another environment allowing customers and partners to use this service. Here the user behavior will be different, but again it must be ensured that the consumers requirements in the new environment will be met.

In general, performance limitations of prospect application contexts cannot be anticipated without proper performance testing. Yet such performance testing requires a detailed understanding of how the provided service APIs are actually used. Therefore, an approach is needed to understand the API consumption of each consumer application, simulate an accurate load behavior of multiple clients and ensure to meet the consumers requirements.

3 Consumer-Driven Contract Testing

Service evolution (like reuse and migration) challenges mutual awareness as well as agreement of consumers and providers. Providers need to be aware of and adjust to the needs of consumers in order to create value but without compromising the evolution of the service system by subsequently introducing 'hidden' coupling. A common pattern to foster evolution in service-oriented systems revolves around *consumer-driven contracts* [9].

Service contracts provide a way of capturing mutual expectations and obligations of providers and consumers as exchangeable artifacts comprising aspects like document schemas, interfaces, conversations, policies and Quality of service

(QoS) characteristics. *Provider contracts* (e.g., WSDL files or OpenAPI[2] specifications) express complete service offerings as initially intended and implemented by their providers. *Consumer contracts*, on the other hand, originate from service clients capturing just those parts of the service offerings that are actually used.

The idea of *consumer-driven contracts (CDCs)* is to synthesize provider contracts out of the complete set of corresponding consumer contracts. On the one hand, CDCs might act as *user stories for services* helping providers to focus their service offers on the actual demand. On the other hand, CDCs might be implemented as *executable contracts* enabling providers to directly test the appropriateness of their current implementation. Moreover, providers could use executable CDCs to anticipate planned changes in the course of service evolution against current expectations and obligations.

The CDC pattern has been implemented by tools like pact[3] and pacto[4] that predominantly focus on executable contracts in the form of functional tests for client and service implementations in isolation.

4 API Performance Contracting

In this section we discuss our approach of refining the CDC pattern for capturing QoS characteristics of application service APIs and support their reuse and migration in different cloud-based application contexts. The first part of this section introduces consumer performance contracts. We then outline how these contracts can be aggregated in order to drive API testing and optimization. Finally, we discuss how performance contracting fits into API lifecycle methodology.

4.1 Consumer Performance Contracts

The key aspect of executing proper performance tests and simulating accurate user behavior is to understand how exactly a consumer makes use of the provided API. Generally, each consumer application corresponds to a set of user scenarios where multiple requests are executed and a specific performance expectation has to be fulfilled. This information needs to be captured and documented in an appropriate way.

We think that consumer contracts (see Sect. 3) can be adopted to fulfill these needs. The idea is to extend consumer contracts with information to understand the load that will be put to production servers and the performance expectations of the users. Each individual client application bound to a service API should provide its own exclusive consumer contract. Following the CDC pattern, load characteristics and consumer expectations of the overall application context are deduced from aggregating individual consumer contracts.

[2] https://openapis.org.
[3] https://github.com/realestate-com-au/pact.
[4] https://thoughtworks.github.io/pacto/.

Besides *organizational metadata*, the contract has to model *business scenarios* that are executed by means of an API. Each scenario must be represented as *set of subsequent API calls*. When modeling virtual users, it is important not only to follow schematic patterns but also consider think times and may even react like frustrated users, abandoning a web session if a response time is excessive [8]. Therefore, *waiting times* between API calls and at the end of each business scenario are added. Furthermore, *dependencies* between business scenarios like causality or probability of follow-up actions need to be modeled. This helps to accurately synthesize user behaviors for performance tests. In the end, the most common business scenarios and important edge cases should be covered. The more complete the set of business scenarios is, the better is the model reflecting client behavior. Beyond modeling client behavior, contracts also need to contain *performance expectations* of a client for each business scenario. This may be represented by average execution time for completing a whole scenario or the maximum response time for each API call. Note, that assertions about the functional correctness such as validation of payload are not part of this contract.

To this point we have described performance contracts as an abstract strategy of modeling client behavior and expectation with respect to the performance of consumed APIs. Such a pattern, like CDC in general, is technology agnostic. In the following sections, we propose a formal representation and tool support for creating and utilizing contracts in the context of application reuse and migration.

4.2 API Contract Aggregation and Evaluation

A provider application is usually not only interested in the usage behavior of one specific consumer but the overall consumption of its API. It must be ensured that the performance expectation of all clients are met simultaneously. Following the approach described so far will result in a set of independent consumer contracts, each of them reflecting a client's individual API usage behavior.

The aggregation of all consumer contracts defines the *consumer-driven performance contract*, including all business scenarios that are used by any of the clients, but also their performance expectations. This information can be utilized to simulate load for performance tests. Virtual users can be derived from these contracts, acting very similar to real users of the production environment. This shifts the creation of usage scenarios to the consumer – the original source.

Executing all business scenarios sequentially would be rather naive and doesn't reflect actual user behaviors. A better approach has been proposed by Abbors et al. with MBPeT, a model-based performance testing tool [1]. MBPeT uses a *probabilistic timed automata (PTA)* [4] to model the behavior of an user. It defines a finite set of locations and transitions that take the PTA from one location to another. Each transition is labeled with a probability value, an action, a clock counter, and a clock reset.

Consumer performance contracts (see Sect. 4.1) can be seen as a declarative approach describing such a PTA. Business scenarios, comprising multiple HTTP requests, can be mapped to a sub-PTA with one or more transitions. Waiting times of performance contracts can be implemented as a clock counter that is

being reset after every transition. Dependencies and possible follow-up actions of business scenarios translate to transitions between sub-PTAs.

Figure 1 shows an example based on an SAPUI5 client of the SAP Relay application (see Sect. 4). Initially, the channel list is requested and the PTA will always reach location $q1$. Users may now either leave the application or select a channel resulting in a location change to $q2$ and multiple HTTP requests retrieving channel details, messages and users. Note that for every location (except the initial one) there must be a transition to the final location $q3$ with a probability greater than 0, since users may leave the application at any time.

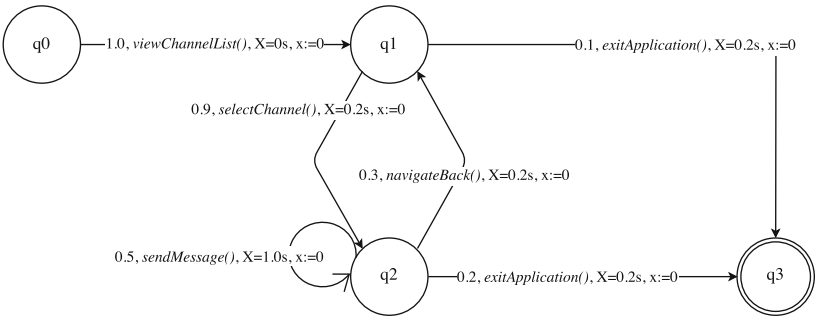


Fig. 1. Example of a probabilistic timed automata

Existing approaches like Kao et al. [5] use an engine to generate performance scripts for performance testing tools out of an abstract model. Accordingly, we use a *contract engine* adopting our pattern of consumer performance contracts by means of a PTA-based testing approach. This tool must be capable of gathering, storing and evaluating consumer contracts, deriving corresponding PTAs and generating load scripts. In particular, the contract engine must simulate the PTA behavior with an appropriate number of users and corresponding sequences of HTTP requests must be exported as a test script.

Since all consumers declare different business transactions and dependencies in their contract, each consumer contract needs to be modeled as separate PTA. This ensures that one virtual user only simulates the load behavior of one specific client. Yet, it is important that there is only one resulting performance script, considering all different consumer types. When simulating virtual users in performance tests, different ratios between consumer types may be realized. This allows reasoning about the performance impact of individual consumers.

With this approach, feedback about the service performance can be provided to the consumer and optimization potentials of the provider application may be revealed. Additional preprocessing and analysis of the contracts may add even more benefit to the consumer-driven performance contract approach. Instead of applying consumer expectations directly, unrealistic performance expectation may be lowered or discarded. Moreover, static analysis by the contract engine may e.g., identify the most common business scenarios, reveal likely performance bottlenecks or detect API misusage before even executing any performance tests.

4.3 Performance Contracting in the API Lifecycle

Within an API lifecycle methodology, using consumer-driven performance contracts can be seen as a continuous process to ensure that performance requirements for a service API are always fulfilled (see Fig. 2).

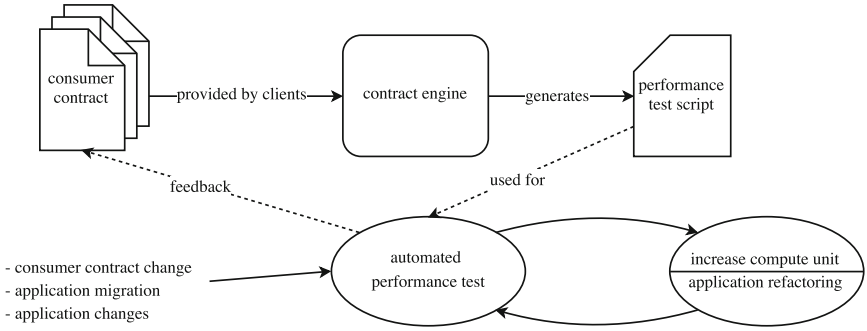


Fig. 2. API lifecycle of performance contracting

Whenever an existing consumer contract is changed or a new client is introduced resulting in another consumer contract, the contract engine gathers the contracts and generates a new aggregated performance script. This script is then used to run automated performance tests. Test results indicate any possible actions that are needed to fulfill the CDC. This may result in a refactoring of the respective endpoints or increasing the compute unit of the provider service.

Furthermore, performance tests may be executed on already deployed applications. Costs could be reduced for existing applications by refactoring/optimizing the API implementation with the goal to meet all requirements, even when running on the next smaller compute unit.

Another important use case for this approach regards the situation when an application has to be migrated to a distinct cloud platform. E.g., in the case of SAP Relay (see Sect. 2) the application is planned to be deployed on a different SAP HCP landscape that provides access not only for SAP employees, but also customers and partners.

5 Related Work

We focus on *adaptive non-functional testing* of *RESTful web service APIs* using *consumer-driven contracts* for *reuse and migration of cloud application services.*

Approaches for adaptive non-functional testing have been proposed in the area of *model-based testing.* E.g., Maâlej et al. have proposed a solution for load testing of WS-BPEL compositions that is based on a timed automata [6]. Abbors et al. have presented the performance testing tool Mbpet [1] and studied

its application in the cloud [2]. We adopt their approach of modeling test cases as PTAs and extend it towards a more complex contract model.

Non-functional testing is also being studied in the area of RESTful web service APIs. Kao et al. have proposed an approach for systematic performance testing of complex REST-based web applications using generated test scripts [5]. Also Zhou et al. presented a model and template-based approach to generate performance test scripts [11]. Similar to these approaches, we propose a contract engine for automated performance test script generation from CDC models.

The field of application reuse and cloud migration increasingly adopts methods of non-functional testing. Strauch et al. identified the need for performance testing while migrating eScience applications to the cloud [10]. Houghtlin et al. describe necessary steps, including performance tests on the different environments, to overcome client concerns around performance when migrating applications to the cloud [7]. Our approach can be used to gather such requirements directly from the consumers and generate the respective performance tests.

6 Conclusion and Future Work

In this paper we have illustrated the challenges of application service provisioning in modern enterprise application landscapes and illustrated them by means of a case study. In particular, we showed how application services that are provided for SAP employees via SAP HCP might be negatively affected when being reused in different application contexts and thereby migrated between cloud-based runtime environments.

In order to support application providers to better anticipate and alleviate the consequences of changing application contexts, we have broadly discussed the novel pattern of *consumer-driven performance contract testing*. More concrete, we have proposed conceptual models for both individual *consumer performance contracts* as well as aggregated *consumer-driven performance contracts* both based on a PTA formalism. Together, they enable automated performance tests by means of generated scripts.

Altogether, the pattern allows for continuously updated performance tests originating from the actual consumer community, which provides very promising conditions for accurate insights in the course of API management.

Currently, we are implementing the contracting toolkit for capturing and sharing individual consumer performance contracts as well as the contract engine to construct aggregated consumer-driven performance contracts for test script generation in the context of SAP HCP.

References

1. Abbors, F., Ahmad, T., Truscan, D., Porres, I.: MBPeT - a model-based performance testing tool. In: Alimohammad, A., Dini, P. (eds.) 4th International Conference on Advances in System Testing and Validation Lifecycle, pp. 1–8. IARIA (2012)
2. Abbors, F., Ahmad, T., Truscan, D., Porres, I.: Model-based performance testing in the cloud using the MBPeT tool. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 423–424. ACM (2013)
3. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)
4. Jurdziński, M., Kwiatkowska, M., Norman, G., Trivedi, A.: Concavely-priced probabilistic timed automata. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 415–430. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04081-8_28
5. Kao, C.H., Lin, C.C., Chen, J.N.: Performance testing framework for rest-based web applications. In: 13th International Conference on Quality Software, pp. 349–354. IEEE, July 2013
6. Maâlej, A.J., Hamza, M., Krichen, M., Jmaïel, M.: Automated significant load testing for WS-BPEL compositions. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 144–153, March 2013
7. Mark Houghtlin, M.E.: Migrating applications to the cloud: assessing performance and response time requirements. Technical report, Cloud Standards Customer Council, October 2014
8. Menasce, D.A.: Load testing of web sites. IEEE Internet Comput. **6**(4), 70–74 (2002)
9. Robinson, I.: Consumer-driven contracts: a service evolution pattern (2006). http://martinfowler.com/articles/consumerDrivenContracts.html
10. Strauch, S., Andrikopoulos, V., Karastoyanova, D., Vukojevic, K.: Migrating e-science applications to the cloud: methodology and evaluation. In: Terzo, O., Mossucca, L. (eds.) Cloud Computing with e-Science Applications, Chap. 5, pp. 89–114. CRC Press/Taylor & Francis, Boca Raton (2015)
11. Zhou, J., Zhou, B., Li, S.: Automated model-based performance testing for PaaS cloud services. In: 2014 IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW), pp. 644–649, July 2014

Patterns and Pattern Languages for SOCC: Use and Discovery, Performance and Conformance of Workflow Engines (PEACE in PATTWORLD)

Preface of PEACE in PATTWORLD

This workshop combines the 1st International Workshop on Performance and Conformance of Workflow Engines (PEaCE) with the 1st International Workshop on Patterns and Pattern Languages for SOCC: Use & Discovery (PATTWORLD).

The goal of PEaCE is to bring the research and industrial practice together, towards topics that are related to the conformance and performance of workflow management systems. Workflow management systems provide platforms for delivering complex service-oriented applications that need to satisfy enterprise-grade quality of service requirements such as dependability and scalability. Benchmarking is an established practice that helps to drive continuous improvement of technology by setting a clear standard and measuring and assessing its performance. For example, transaction processing benchmarks have been introduced since a long time and over decades they have been instrumental to achieve an enormous performance improvement of database technology, e.g., with the TPC family of benchmarks. Conversely, benchmarks for service oriented computing in general and workflow management systems in particular have started to appear only recently and there is no currently accepted standard benchmark. Any vendor can claim that their product is standard compliant while still implementing different subsets or interpretations of the standard, since there is no certification authority for both of the most popular and widespread business process languages standards BPEL and BPMN. The different interpretations and implementations of the standards led and still lead to vendor lock-ins as porting a standard compliant workflow involves too much effort, and sometimes is not possible at all—effectively killing the standard inherent argument for portability. Benchmarks for standard conformance have not been around for long in the area of WfMSs, and still lack scientific foundation in their creation and execution.

The context of PATTWORLD are patterns, which have emerged in several IT domains as lingua franca to document proven solutions for frequently reoccurring problems. Especially in the domains of Service-Oriented Computing and Cloud Computing, the pattern concept is recently used by academia and industry to capture and use proven knowledge about efficiently designing, building, and managing IT systems. Well-established pattern languages, e.g., Enterprise Integration Patterns and Messaging Patterns, complement these works, thereby, providing a huge knowledge base for the development of future IT systems. In addition, new evolutions, such as the Internet of Things and the emerging trend of microservices, provide promising topics for the development of new pattern languages and pattern usage concepts. Practitioners are desperately looking for solutions for developing applications in these domains. Especially discovering, using, and structuring patterns are heavily researched fields to create and maintain a knowledge base that can be used efficiently by humans. Moreover, also concepts for automating the application of patterns to individual use cases gain more and more attention in very different IT domains, e.g., automating patterns for the management of Cloud-based applications. A plethora of websites and blogs prove the strong interest in finding solutions to build proper microservices but lack a

systematic approach. Therefore, finding new ways and general concepts to increase the efficiency of discovering and using patterns is of great interest, which possibly affects multiple different domains at once.

The workshop opened with the keynote of Beniamino di Martino on “Semantics and Patterns to support MultiCloud Applications' Portability and Cloud Services Orchestration and Composition”. We are grateful for his insightful presentation focusing on Cloud portability which set the tone for the rest of the workshop. The first presentation was by Vincenzo Ferme on “Workflow Engine Performance Benchmarking with BenchFlow,” summarizing the results of the BenchFlow project and presenting early results on the performance evaluation of open-source workflow engines using workflow patterns as workloads for micro-benchmarks. These results are being embedded into the “Interactive Dashboard for Workflow Engine Benchmarks” presented by David Bimamisa, Mathias Müller, Simon Harrer, and Guido Wirtz. In addition the dashboard gives an interactive view over the actual compatibility of different versions of different workflow engines with the BPMN 2.0 standard. This not only allows to track how the standard compliance of these engines has evolved over time, but also to see which BPMN features are most supported in practice. The session on Performance and Conformance was concluded by the paper on “A distributed cross-layer monitoring system based on QoS metrics models” by Damianos Metallidis, Kyriakos Kritikos, Chrysostomos Zeginis, and Dimitris Plexousakis. The work proposes to gather monitoring data from multiple layers (Workflow, Service and Infrastructure) with a cross-layer quality model. The session on “Patterns and Pattern Languages” was opened by the paper on “Patterns for Workflow Engine Benchmarking” by Simon Harrer, Oliver Kopp, and Jörg Lenhard. The patterns address important design challenges of workflow engine benchmarking and cover how to design suitable tests, how to interface the engines under test with the benchmarking procedure, as well as how to validate the results of the benchmark. The last paper was on “Patterns in HCI – A Discussion of Lessons Learned” by Alexander G. Mirnig, Artur Lupp, and Manfred Tscheligi. The authors reflect upon their pattern finding and writing activities in the user experience design for the automotive human-computer interaction domain. We wish that you enjoy reading the papers as much as we did and look forward to next year's edition of the workshop.

Oliver Kopp
Eva Kühn
Jörg Lenhard
Frank Leymann
Cesare Pautasso
Guido Wirtz

Organization

Workshop Organizers

Oliver Kopp	IPVS, University of Stuttgart, Germany
Eva Kühn	TU Vienna, Austria
Jörg Lenhard	Karlstad University, Sweden
Frank Leymann	IAAS, University of Stuttgart, Germany
Cesare Pautasso	University of Lugano, Switzerland
Guido Wirtz	University of Bamberg, Germany

PATTWORLD Program Committee

Uwe Breitenbücher	University of Stuttgart, Germany
Manuel Wimmer	TU Vienna, Austria
Oliver Kopp	University of Stuttgart, Germany
Schahram Dustdar	TU Vienna, Austria
Massimo Villari	University of Messina, Italy
Andreas Metzger	University of Duisburg-Essen, Germany
Florian Daniel	Politechnico Milano, Italy
Kostas Magoutis	University of Ioannina Greece
Uwe Zdun	University of Vienna, Austria
Christian Kohls	TH Köln, Germany
Winfried Lammersdorf	University of Hamburg, Germany
Simon Moser	IBM, Germany
Srinath Perera	WSO2, Sri Lanka
Willem-Jan van Heuvel	University of Tilburg, Netherlands
Cesare Pautasso	University of Lugano, Switzerland
Elisbetta di Nitto	Politechnico Milano, Italy
Beniamino di Martino	University of Naples, Italy
Antonio Brogi	University of Pisa, Italy

PEACE Program Committee

Andre van Hoorn	University of Stuttgart, Germany
Andreas Rogge-Solti	WU Vienna, Austria
Barbara Pernici	Politecnico di Milano, Italy
Christoph Hochreiner	TU Vienna, Austria
Claudio Di Ciccio	WU Vienna, Austria
Dimitris Plexousakis	University of Crete, Greece
Fabio Casati	University of Trento, Italy

Ingo Weber	University of New South Wales, Australia
Marigianna Skouradaki	University of Stuttgart, Germany
Matthias Geiger	University of Bamberg, Germany
Matthias Weidlich	HU Berlin, Germany
Matthias Weske	Hasso Plattner Institute, Potsdam, Germany
Patrick Delfmann	University of Münster, Germany
Uwe Breitenbücher	University of Stuttgart, Germany
Vincenzo Ferme	University of Lugano, Switzerland
Vinod Muthusamy	Thomas J. Watson Research Center, Yorktown Heights, NY USA

Patterns for Workflow Engine Benchmarking

Simon Harrer[1(✉)], Oliver Kopp[2], and Jörg Lenhard[3]

[1] Distributed Systems Group, University of Bamberg, Bamberg, Germany
simon.harrer@uni-bamberg.de
[2] Institute for Parallel and Distributed Systems,
University of Stuttgart, Stuttgart, Germany
oliver.kopp@informatik.uni-stuttgart.de
[3] Department of Mathematics and Computer Science,
Karlstad University, Karlstad, Sweden
joerg.lenhard@kau.se

Abstract. Workflow engines are frequently used in the service-oriented and cloud computing domains. Since engines have significant impact on the quality of service provided by hosted applications, it is desirable to compare and select the most appropriate engine for a given task. To enable such a comparison, approaches for benchmarking workflow engines have emerged. Although these approaches deal with different quality properties, such as performance or standard conformance, they face many reoccurring problems during the design and implementation phase, which they solve in similar ways. In this paper, we describe such common solutions to reoccurring problems in the area of workflow engine benchmarking as patterns. Our aim is to present pattern candidates that help benchmark authors to design and implement proper and valid workflow engine benchmarks and benchmarking tools.

Keywords: Patterns · Workflow engine · Benchmarking

1 Introduction

An established part of the field of service-oriented computing is the construction of composite services on the basis of message exchanges between lower-level services [21]. This composition is often achieved by capturing the data- and control-flow between message exchanges of several services in a workflow [22]. The workflow is subsequently deployed on a *workflow engine*, which provides the middleware execution platform, context and cross-cutting functionality, message correlation, and many other features to the hosted workflow. Today, several standards for workflow definition and a multitude of engines have emerged, including implementations by multi-national middleware vendors, open source solutions, research prototypes, and even cloud-based engines. The range of solutions makes it important to the user to compare existing engines with the aim of selecting the best engine for her purpose. However, engines are highly complex products, resulting in an equally complex selection problem [9]. To address this problem,

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 151–163, 2018.
https://doi.org/10.1007/978-3-319-72125-5_12

workflow engine benchmarking approaches have emerged [3,6]. Several research groups are developing different benchmarking approaches and tools that target varying quality properties of workflow engines, such as performance [3] or standard conformance [6].

Currently, approaches for workflow engine benchmarking are being developed in parallel. Their authors often face the same problems, regardless of the actual property that lies in the focus of the benchmark. Such common problems are, for instance, how to identify suitable tests or workloads for engine benchmarks, or how to ensure the correctness of test implementations. Moreover, solutions to such common problems are often similar, leading to the unfortunate situation that multiple groups invest significant effort to solve the same problem and re-implement the same solution. Since proven solutions to reoccurring problems exist and can be inferred from existing engine benchmarks, it is possible to capture these solution as *patterns* [1,5]. By describing such solutions as patterns, it should be possible to reduce the effort for implementing new workflow engine benchmarks and also to ease the communication among benchmark authors through a common vocabulary.

This paper is a first attempt to provide pattern candidates within the domain of workflow engine benchmarking. Working on workflow engine benchmarks and tools for several years, we are confident that the presented patterns can help the authors of future benchmarks. It should be possible to extend and refine the proposed pattern candidates in the future. In the following, we develop the pattern candidates by describing the participants and challenges in workflow engine benchmarking, for which we propose a number of solutions. These solutions should provide guidelines to future benchmark authors when facing the same challenge.

The paper is structured as follows. First, we present related work in Sect. 2, providing the background for describing the participants and challenges in workflow engine benchmarking in Sect. 3. In Sect. 4, pattern candidates are presented which act as a set of alternative and competing solutions to the challenges outlined in Sect. 3. The work is concluded with an outlook on future work in Sect. 5.

2 Related Work

Work related to this paper can be roughly divided into three areas: (i) The definition and usage of patterns, (ii) work on workflows and workflow engines, and (iii) work on benchmarking and benchmarking workflow engines in particular.

The notion of patterns originated from the field of architecture [1], where patterns were used to describe reoccurring structures in buildings. Years later, the idea to describe reoccurring structures in the design of software in the form of patterns [5] had a huge impact on software development. Since then, patterns have been applied in many areas and contexts, and a multitude of pattern catalogs and languages have been published. Workflow engine benchmarking is an area, where, to the best of our knowledge, patterns are still lacking. The huge momentum in the development of pattern languages, has also led to work that theorizes

on pattern structure [17,18] and how to write a pattern [19]. Here, we build on these works to specify our pattern candidates. Meszaros and Doble [19] propose *name*, *problem*, *context*, *forces*, and *solution* as the mandatory elements of patterns. *Examples* and *relations* are considered as optional elements. In our work, we use *name*, *problem*, *solution*, *relations*, and *example* as elements for pattern description. Our paper aims to provide a list of pattern candidates we discovered in our work with workflow engine benchmarking. Due to space limitations, we excluded *context* and *forces* from the presentation of our pattern candidates.

Workflows and workflow engines, or more abstract, process-aware information systems [25], are commonly used in the service-oriented computing domain to orchestrate services [22]. In short, a workflow is the machine-readable and executable representation of a business process in whole or part and a workflow engine is the software runtime environment that manages and controls the execution of workflow instances [27]. Today, two language standards are predominantly used for workflow specification and execution. These are the Web Services Business Process Execution Language (BPEL) [20] and the Business Process Model and Notation (BPMN) [16].

Benchmarks are an important tool in computer science that is needed to compare and analyze the quality provided by software systems [15]. Many aspects of software can be benchmarked, but often the focus resides on performance-related aspects, such as latency or throughput. When it comes to workflow engines, two major aspects have been in the spotlight. As indicated above, one of these is performance [2,3]. The second aspect is standard conformance, which reflects the fact that workflow engines are often standards-based products [7,10]. Benchmarking approaches for both aspects exist for both languages mentioned in the previous paragraph, for BPEL [2,10] and BPMN [3,7]. The BPEL/BPMN Engine Test System betsy[1] and BenchFlow[2] are implementations of these benchmarking approaches.

3 Problems in Workflow Engine Benchmarking

In the following subsection, we clarify the reoccurring challenges one faces when building a benchmark for workflow engines. These challenges are the crucial sources and motivation for gathering workflow engine benchmarking patterns.

3.1 Big Picture

In the big picture of workflow engine benchmarking, there are four elements: *tests*, the *engines* to be tested, the *benchmarking procedure*, and the benchmark *results*.

When a benchmark is conducted, *tests* are used to specify requirements or desired behavior of the engines under test. Next to the tests, these *engines* are the second input to the benchmark. They are the objects of study that are to be

[1] https://github.com/uniba-dsg/betsy.
[2] https://github.com/benchflow/benchflow.

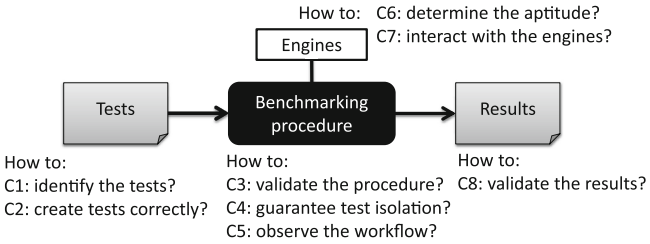


Fig. 1. Big Picture of Workflow Engine Benchmarking

evaluated in a comparable fashion. The *benchmarking procedure* is the tool that instruments both, engines and tests. It evaluates the first using the latter and produces benchmark *results*. These results should be constructed in an easily comprehensible fashion to allow for a straight-forward interpretation (Fig. 1).

3.2 Challenges

During workflow engine benchmarking, a number of challenges arise related to each element listed in Sect. 3.1. These challenges are non-trivial and reoccurring problems that need to be solved for every benchmark, hence, they are the problems for which we propose patterns as a solution in this paper. In total, we identified eight challenges, numbered from C1 to C8, which we present in the following.

Regarding the tests, the major issues are how to *identify the tests (C1)* and *create tests correctly (C2)*. The tests should be suitable and representative of a realistic usage scenario. If this is not the case, the results produced by the benchmark are of no use. Since realistic tests can be non-trivial, it is important to ensure that they are free of issues, since even minor issues could have considerable impact on the benchmark results.

Major challenges regarding the benchmarking procedure are how to *validate the procedure (C3)*, *guarantee test isolation (C4)*, and *observe the workflow (C5)*. As for the tests, quality assurance needs to be in place to make sure that there are no errors in the benchmarking procedure that might have an impact on the benchmark results. As realistic test sets might be large, it is important to make sure that tests can be executed independently, regardless of the execution order, and regardless of whether execution takes place sequentially or in parallel. Finally, a mechanism needs to be in place that helps to identify how and if the benchmark and singular tests are progressing. Since a benchmark might push an engine to its limits, it can easily be the case that an engine fails to make progress during execution, which needs to be detected and acted upon.

Regarding the engines, the major issues are how to *determine the aptitude (C6)* of an engine and how to *interact with the engines (C7)*. The sixth challenge concerns the ability of the engine to participate in the first place. The engine needs to be in normal working mode at the start of the benchmark, otherwise

meaningful results are unlikely. C7 concerns the ability to control and monitor the engine during execution. Test execution requires the evaluation of assertions or observation of behavior, so it is necessary that the engine has facilities in place that allow to communicate its state to the outside.

Finally, regarding the results, the major issues are how to *validate the results (C8)*. Although the benchmark procedure might have worked without problems, it is possible that there are errors in the conversion of raw benchmark data to interpretable results. Therefore, quality assurance is needed to make sure that the results are actually correct.

4 Workflow Engine Benchmarking Pattern Catalog

In the following section, we describe our candidates for workflow engine benchmarking patterns. The candidates are structured into several categories that mirror the elements of a benchmark from Sect. 3.1 and are based on the challenges described in the previous section. *Test patterns* concern the identification and quality assurance of test cases for a benchmark, and *procedure patterns* group patterns for automating the benchmark environment. *Engine patterns* describe ways to instrument workflow engines for using them in a benchmark. Finally, we present *results patterns*, which provide solutions for the validation of result data. This section concludes with a short discussion of the patterns.

For each pattern, we provide a unique *name* and list the *problems* it addresses, which directly corresponds to the challenges from Sect. 3.2. Furthermore, we describe the *solution* to said problem, which outlines what the pattern does, in an abstract form, whereas the *example* describes its known usage and *relations* to other patterns.

Test Patterns: To *identify the tests (C1)*, one can determine the constructs of a language and apply *Configuration Permutation (P1)* or use *Reoccurring Constructs (P2)* to identify the most used constructs and their configuration. To *create tests correctly (C2)*, one can derive tests using *Stub Extension (P3)* or *Mutated Existing Test (P4)*, which both ensure a certain degree of correctness. Applying *Open Sourcing (P5)*, *Expert Review (P6)*, and *Automatic Static Analysis (P7)* helps to strengthen the degree of correctness further. What is more, the latter patterns can be applied independently of each other.

Name. Configuration Permutation (P1)
Problem. Identify the tests (C1)
Solution. Identify a construct. Permutate all configurations of a construct. Each permutation is a test.
Example. This pattern was applied in betsy for BPEL and BPMN standard conformance tests. For instance, in BPMN, there is the construct *exclusive gateway* which can be configured in three ways resulting in three tests: (i) *standard* with all outgoing sequence flows having conditions, (ii) exclusive gateway with a sequence flow without a condition and marked as *default*, and (iii) one as a *mixed* gateway with both branching and merging capabilities.

Name. Reoccurring Constructs (P2)
Problem. Identify the tests (C1)
Solution. Gather a large corpus of workflows. Identify the reoccurring elements in these workflows. Tests are created based on the most important (i.e., reoccurring) elements.
Example. This pattern was applied in betsy for BPEL and BPMN expressiveness tests as the test suite is based on the workflow control-flow patterns [26,28] which are created from analyzing multiple workflow management systems. In BenchFlow, a large corpus of workflows is used to construct workloads for performance tests [23,24].

Name. Stub Extension (P3)
Problem. Create tests correctly (C2)
Solution. Use a workflow stub, a minimal workflow, which is extended for all tests, so that the extension contains solely the feature under test. The stub itself provides extension points, where the feature under test can be put. The rest is minimal overhead required to observe the feature under test. This way, all tests follow the same structure, and when looking at the difference between the test and the stub, the feature under test can be easily identified.
Example. This pattern was applied in betsy for both, BPEL and BPMN.
Related. If the stub is fully functional, it can act as an Aptitude Test (P8).

Name. Mutated Existing Test (P4)
Problem. Create tests correctly (C2)
Solution. Instead of starting from scratch, use correct tests and modify them by introducing a mutation [12]. This is especially useful for creating tests for faulty conditions: An existing test is mutated by injecting a single isolated fault, to see if a feature works correctly even in the face of errors [13].
Example. The pattern was applied in betsy for creating erroneous workflows that have to rejected upon deployment for both BPEL and BPMN. In addition, it was applied to create a test suite for determining robustness [12].
Relations. Similar to Stub Extension (P3) as the workflow model of another test as the basis for a new workflow test.

Name. Open Sourcing (P5)
Problem. Create tests correctly (C2), validate the procedure (C3), and validate the results (C8)
Solution. Open source tests, procedure, and results, and put it under the scrutiny of the public. Public availability can help to find errors the original authors did not find. Also, this can help to build a community for the benchmark.
Example. Both betsy and benchflow are open source. In case of betsy, this has lead to contributions by experts and also engine vendors.
Relations. May result in Expert Review (P6).

Name. Expert Review (P6)
Problem. Create tests correctly (C2), validate the procedure (C3), and validate the results (C8)
Solution. Ask experts to review the benchmark artifacts. Experts can be domain experts, engine developers, or benchmark engineers.
Example. For betsy, the maintainers of Apache ODE and bpel-g helped to improve the test cases through their feedback, looking at the results and checking why the behavior of their engine was different from what they expected.

Name. Automatic Static Analysis (P7)
Problem. Create tests correctly (C2), validate the procedure (C3), and validate the results (C8)
Solution. Create static analysis checks which detect mistakes automatically. As most workflow languages are XML-based, an XML well-formedness check as well as schema validation with the XSD of the workflow language is straightforward. If possible, apply additional static analysis based on workflow language rules and best practices.
Example. The pattern was applied in betsy by checking the correctness of workflows regarding naming conventions, XML well-formedness, XSD validity regarding the workflow language schema, and even more sophisticated static analysis rules with BPELlint[3] and BPMNspector[4].

Benchmarking Procedure Patterns: To *validate the procedure (C3)*, one can apply *Open Sourcing (P5)*, *Expert Review (P6)*, and *Aptitude Test (P8)*. *Reinstallation (P9)*, *Virtual Machines (P10)*, and *Containers (P11)* can be applied to *guarantee test isolation (C4)*. To *observe the workflow (C5)*, *Message Evaluation (P12)*, *Partner-based Message Evaluation (P13)*, *Execution Trace Evaluation (P14)*, *Engine API Evaluation (P15)*, *Concurrency Detection (P16)*, and *Detailed Logs (P17)* are applicable.

Name. Aptitude Test (P8)
Problem. Validate the procedure (C3) and determine the aptitude (C6)
Solution. Define an aptitude test as a minimal requirement for participation in the benchmark. An engine must pass this test. The test should check the minimal amount of features required.
Example. In betsy, there are two aptitude tests, one for BPEL named *Sequence*, containing a receive-assign-reply triplet (see Message Evaluation (P12)), and one for BPMN, named *SequenceFlow*, containing a start and end event, with corresponding script tasks to allow to observe the events (see Execution Trace Evaluation (P14)), connected through sequence flows.
Relations. Can be used as a stub for Stub Extension (P3).

Name. Reinstallation (P9)
Problem. Guarantee test isolation (C4)

[3] https://github.com/uniba-dsg/BPELlint.
[4] http://bpmnspector.org/.

Solution. Install and start the engine anew for each test case, providing a fresh engine instance. Although a reinstallation can consume a lot of time, it ensures that one test case cannot interfere with another one.
Example. In betsy, this is the default mode.
Relations. Virtual Machines (P10) and Containers (P11) are alternatives.

Name. Virtual Machines (P10)
Problem. Guarantee test isolation (C4)
Solution. Create a virtual machine with a snapshot of a running engine upfront. This may require some time and effort once per engine. But with a snapshot in place, each test can be executed in isolation. The snapshot can easily be restored before each test and be discarded afterwards, resulting in test isolation with a low temporal overhead. However, for virtual machines, there is typically a substantial RAM and HDD overhead.
Example. Since 2014, betsy also supports this pattern [14].
Relations. Reinstallation (P9) and Containers (P11) are alternatives.

Name. Containers (P11)
Problem. Guarantee test isolation (C4)
Solution. Create an image with the engine already installed and configured. Create a new container for each test and discard the container afterwards, effectively ensuring test isolation. This is similar to Virtual Machines (P10), but with considerably less overhead. At this point in time, however, support for RAM snapshots of containers is not existing, but HDD snapshots are available.
Example. This pattern is used in both, betsy [8] and BenchFlow [4].
Relations. Reinstallation (P9) and Virtual Machines (P10) are alternatives

Name. Message Evaluation (P12)
Problem. Observe the workflow (C5)
Solution. Send messages to the workflow and compare responses with an expected response. Use small interfaces with only few methods to keep different message types and possibilities low.
Example. Betsy communicates with BPEL instances only through four different SOAP messages and observes the behavior by checking the responses.
Relations. Partner-based Message Evaluation (P13) builds upon this pattern. Execution Trace Evaluation (P14) and Engine API Evaluation (P15) are alternatives.

Name. Partner-based Message Evaluation (P13)
Problem. Observe the workflow (C5)
Solution. The workflow under test sends messages to an external service which the benchmarking system controls. The calling pattern of the service can be checked and compared to the expected interaction.
Example. In betsy, this pattern is used to mock any partner service a BPEL workflow is required to communicate with. Moreover, concurrency detection was implemented with a mocked partner service as well.

Relations. This pattern is an extension of Message Evaluation (P12). Alternatives are Execution Trace Evaluation (P14) and Engine API Evaluation (P15). Concurrency Detection (P16) can be implemented using this pattern.

Name. Execution Trace Evaluation (P14)
Problem. Observe the workflow (C5)
Solution. The workflow writes log traces to the disk. The benchmarking framework then reads the log traces and compares them with expected ones. Use a small set of different standardized log traces. One can even inspect Detailed Logs (P17) and convert log statements to log traces.
Example. This pattern is used in process mining, but also in betsy for observing the behavior of BPMN workflows, as Message Evaluation (P12) does not work because of the lacking support for sending and receiving messages. In script tasks, log traces are written to a log file. Moreover, engine specific logs are checked and additional log traces are created based on them. This is useful for conditions like the detection of whether a workflow did exit correctly.
Relations. Partner-based Message Evaluation (P13), Message Evaluation (P12), Engine API Evaluation (P15), Concurrency Detection (P16), and Detailed Logs (P17).

Name. Engine API Evaluation (P15)
Problem. Observe the workflow (C5)
Solution. Use the API provided by the engine to query the deployment state of the workflow model and the current state as well as the history of specific workflow instances. As this is engine dependent, it profits from applying Engine Layer Abstraction (P18) as well.
Example. In both betsy and BenchFlow, the BPMN engines are queried about their deployment and final states.
Relations. Alternative to Partner-based Message Evaluation (P13), Message Evaluation (P12), and Execution Trace Evaluation (P14), but works fine together with Engine Layer Abstraction (P18).

Name. Concurrency Detection (P16)
Problem. Observe the workflow (C5)
Solution. Identify the parallel branches in the workflow under test. Upon entering and exiting each branch, a timestamp has to be stored alongside a branch identifier at runtime. If the enter-exit pairs of parallel branches overlap, real concurrency has been detected. The concurrency traces can either be tracked through an an external service or a log file.
Example. Betsy applies this pattern relying on Partner-based Message Evaluation (P13) for BPEL and Execution Trace Evaluation (P14) with a separate concurrency detection log file for BPMN.
Relations. Can be used either with Partner-based Message Evaluation (P13) or Execution Trace Evaluation (P14).

Name. Detailed Logs (P17)
Problem. Observe the workflow (C5)

Solution. Configure the engine to use verbose logging. Otherwise, it might not be possible to observe everything that is important regarding the state of a workflow.
Example. In betsy, detailed logs are enabled for several engines by replacing the log configuration file with a more verbose one.
Relations. Execution Trace Evaluation (P14) and Concurrency Detection (P16).

Engine Patterns: To *determine the aptitude (C6)*, one can apply the *Aptitude Test (P8). Engine Layer Abstraction (P18)*, *Failable Timed Action (P19)*, *Timeout Calibration (P20)* and *Detailed Logs (P17)* can be used to *interact with the engines (C7)*.

Name. Engine Layer Abstraction (P18)
Problem. Interact with the engines (C7)
Solution. Create an abstract layer which (a) converts engine independent artifacts to engine dependent ones and vice versa, and (b) provides uniform methods to interact with each engine. This handles converting engine specific logs to engine independent log traces, installation, deployment, starting, and other engine specific assertions such as how to behave after an abortion of a workflow.
Example. The Uniform BPEL Management Layer (UBML) [11] has been extracted from betsy and it is an engine independent layer to (un)install, start, and stop the engine as well as to deploy workflows and collect log file. The engine adapters of this layer heavily rely on Failable Timed Action (P19), Timeout Calibration (P20), and Detailed Logs (P17).
Relations. Can rely upon Failable Timed Action (P19), Timeout Calibration (P20), and Detailed Logs (P17). In addition, eases Reinstallation (P9).

Name. Failable Timed Action (P19)
Problem. Interact with the engines (C7)
Solution. The test system executes a specified action. Then it waits for a specific period during which success and failure conditions are checked every X milliseconds. The action fails if time is exceeded or if failure condition is met. It succeeds if success condition is met within the specific period.
Example. As most engines do not support a synchronous API, betsy needs to rely on Failable Timed Action (P19). The act of deploying a workflow often involves copying the artifact to a specific location on the file system, after which the engine deploys it automatically, and then evaluating success through log inspection (see Detailed Logs (P17)).
Relations. May require Detailed Logs (P17), should be used with Timeout Calibration (P20).

Name. Timeout Calibration (P20)
Problem. Interact with the engines (C7) and validate the results (C8)

Solution. Before an actual machine is used for benchmarking, calibrate the timeouts that are required in the tests itself, in a Failable Timed Action (P19), or for Reinstallation (P9). The calibration of timeouts in the tests is necessary, in case a test produces non-deterministic results depending on different timeout settings.
Example. Betsy implements a mechanism using the Aptitude Test (P8) to calibrate typical timeout values with a security range.
Relations. Ensures that the timeouts in Reinstallation (P9) and Failable Timed Action (P19) are suitable.

Results Patterns: To *validate the results (C8)*, one can apply *Open Sourcing (P5)*, *Expert Review (P6)*, *Aptitude Test (P8)*, *Timeout Calibration (P20)* and *Single Success (P21)*.

Name. Single Success (P21)
Problem. Validate the results (C8) and create tests correctly (C2)
Solution. Compare the benchmarking results of one test between engines. If the test does not succeed on at least one engine, it is necessary to investigate the test itself, since the test or the testing procedure might be broken.
Example. In betsy, this pattern was applied multiple times to detect issues in the tests, the procedure and the interaction with engines.

Discussion: Most of the patterns are only used for a single challenge. The pattern Detailed Logs (P17) can be applied twice, once to observe the workflow (C5) and to interact with the engines (C7). Three patterns, namely, Open Sourcing (P5), Expert Review (P6), and Aptitude Test (P8), are applicable to three challenges each. Open Sourcing (P5) and Expert Review (P6) are more general patterns not specific to workflow benchmarking, but they helped us tremendously with quality assurance, namely, to create tests correctly (C2), validate the procedure (C3), and validate the results (C8). The Aptitude Test (P8) is more specific to the benchmarking domain, and can be used to validate the procedure (C3), determine the aptitude (C6), and validate the results (C8).

5 Conclusion and Future Work

In this paper, we introduced pattern candidates for workflow engine benchmarking. We identified the central elements in this domain, as well as major challenges one faces when building a benchmark. The patterns we proposed are categorized, according to these aspects, into test, procedure, engine, and results patterns.

In the future, we want to improve and refine the pattern candidates with domain experts, i.e., the authors of other workflow engine benchmarks or workflow engines. In the long term, these patterns could provide a useful basis for future benchmark authors and could help to establish a common vocabulary in the domain. Last, we see potential in generalizing these specific patterns for benchmarking workflow engines to more generic patterns for benchmarking other standard-based software or even to patterns for benchmarking in general.

References

1. Alexander, C.: A Pattern Language, August 1978
2. Bianculli, D., Binder, W., Drago, M.L.: SOABench: performance evaluation of service-oriented middleware made easy. In: ICSE, pp. 301–302. ACM (2010)
3. Ferme, V., Ivanchikj, A., Pautasso, C.: A framework for benchmarking BPMN 2.0 workflow management systems. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 251–259. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_18
4. Ferme, V., Ivanchikj, A., Pautasso, C., Skouradaki, M., Leymann, F.: A container-centric methodology for benchmarking workflow management systems. In: CLOSER (2016)
5. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Amsterdam (1995)
6. Geiger, M., Harrer, S., Lenhard, J.: Process engine benchmarking with betsy - current status and future directions. In: ZEUS, pp. 37–44, January 2016
7. Geiger, M., Harrer, S., Lenhard, J., Casar, M., Vorndran, A., Wirtz, G.: BPMN conformance in open source engines. In: SOSE, March 2015
8. Geiger, M., Harrer, S., Lenhard, J., Wirtz, G.: On the evolution of BPMN 2.0 support and implementation. In: SOSE, pp. 120–128, March 2016
9. Harrer, S.: Process engine selection support. In: Meersman, R., et al. (eds.) OTM 2014. LNCS, vol. 8842, pp. 18–22. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45550-0_3
10. Harrer, S., Lenhard, J., Wirtz, G.: BPEL conformance in open source engines. In: SOCA, pp. 237–244. IEEE, December 2012
11. Harrer, S., Lenhard, J., Wirtz, G., van Lessen, T.: Towards uniform BPEL engine management in the cloud. In: CloudCycle, LNI. GI e.V., September 2014
12. Harrer, S., Nizamic, F., Wirtz, G., Lazovik, A.: Towards a robustness evaluation framework for BPEL engines. In: SOCA, pp. 199–206. IEEE, November 2014
13. Harrer, S., Preißinger, C., Wirtz, G.: BPEL conformance in open source engines: the case of static analysis. In: SOCA, pp. 33–40. IEEE, November 2014
14. Harrer, S., Röck, C., Wirtz, G.: Automated and isolated tests for complex middleware products: the case of BPEL engines. In: ICSTW (2014)
15. Huppler, K.: The art of building a good benchmark. In: Nambiar, R., Poess, M. (eds.) TPCTC 2009. LNCS, vol. 5895, pp. 18–30. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10424-4_3
16. ISO/IEC. ISO/IEC 19510:2013 - Information technology - Object Management Group Business Process Model and Notation, November 2013. v2.0.2
17. Kohls, C.: The structure of patterns. In: PLOP. ACM (2010)
18. Kohls, C.: The structure of patterns - part ii - qualities. In: PLOP. ACM (2011)
19. Meszaros, G., Doble, J.: A pattern language for pattern writing. Pattern Lang. Program Des. **3**, 529–574 (1998)
20. OASIS. Web Services Business Process Execution Language, April 2007. v2.0
21. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. IJCIS **17**(2), 223–255 (2008)
22. Peltz, C.: Web services orchestration and choreography. IEEE Comput. **36**(10), 46–52 (2003)
23. 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

24. Skouradaki, M., Roller, D.H., Leymann, F., Ferme, V., Pautasso, C.: On the road to benchmarking BPMN 2.0 workflow engines. In ICPE. ACM (2015)
25. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**, 1–37 (2013)
26. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(1), 5–51 (2003)
27. WfMC. The Workflow Reference Model, January 1995. v1.1
28. Wohed, P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives (revised version). BPM Center Report BPM-06-17 (2006)

Patterns in HCI – A Discussion of Lessons Learned

Alexander G. Mirnig(✉), Artur Lupp, and Manfred Tscheligi

Christian Doppler Laboratory "Contextual Interfaces", Center for Human-Computer Interaction, Department of Computer Sciences, University of Salzburg, 5020 Salzburg, Austria
{alexander.mirnig,artur.lupp,manfred.tscheligi}@sbg.ac.at
http://hci.sbg.ac.at

Abstract. Patterns are a tool to capture best practices and solutions to reoccurring problems in certain domains. Patterns allow capturing very specific and context-dependent solutions for re-use by individuals of different levels of expertise and are, therefore, a valuable supplement to guidelines and other means of general guidance. In our work paper, we applied the pattern approach to User Experience (UX) design in automotive human-computer interaction. Our approach combines findings from scientific studies and industry stakeholder's knowledge and distills them into automotive patterns for use by in-vehicle system designers. Over the course of several years of pattern finding and writing workshops, we encountered a number of issues, solutions, and lessons learned, which we want to share with the pattern community. These encompass relevant problem finding, language use and referencing in multidisciplinary contexts, and the problem of adequately integrating purely scientific findings into the rather practically oriented patterns concept. We discuss each point in detail and provide our internal solutions and lessons learned for each of these, together with a summary of problems, which are still to be solved and future work potentials.

Keywords: Human-Computer-Interaction · Cross-discipline Patterns · Patterns in practice · Lessons learned

1 Introduction

Patterns are a tool and method to capture working solutions to reoccurring problems, embedded in the context they occurr(ed) in. While they have their beginnings in architecture, in contemporary literature patterns are mostly used in software engineering, although they are employed in other disciplines as well. Human-Computer-Interaction (HCI) is among those disciplines. The need to capture problem solutions with particular regard to contextual factors and reusability is very present in HCI, due to its interdisciplinarity, close ties to industry advancements, and general rapidly progressing nature.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 164–175, 2018.
https://doi.org/10.1007/978-3-319-72125-5_13

While pattern approaches are well-developed for the software engineering side of informatics, other pattern approaches vary greatly in level of detail and quality, due to very little existing general literature on patterns and pattern approaches. This can make it difficult to put the concept of patterns into practice without prior experience. Using and generating patterns, much like any other tool, takes practice and knowledge until a high level of quality is achieved. Some of these problems can be handled at a theoretical level, although many are related to lower level, i.e. practical, problems, such as the number of individuals to involve in a pattern iteration, whether iterations should include a mix of individuals familiar and unfamiliar with the pattern (fresh eyes vs. expert eyes), differences in language and reference style (the person who writes a pattern is not always in the group of individuals who will later use the pattern), a.s.o.

In this paper, we discuss a number of practical problems we encountered during our pattern generation activities, together with the lessons learned and strategies we developed to deal with these issues. After an overview of related work regarding general literature on pattern mining and writing, we present a list of five issues. Each issue is briefly outlined, after which we present and discuss the strategies we used to deal with each of these issues. At the end, we summarize these strategies as five lessons learned. This contribution should foster discussion regarding practical aspects of pattern generation as well as provide some preliminary solutions for such problems others might already have encountered when working with patterns. The solutions or lessons learned we provide are not necessarily specific to HCI and are intended for anyone who employs patterns for documenting problem solutions.

2 Related Work

Patterns, as a tool to capture problem solutions, were first introduced by Christopher Alexander specifically for capturing solutions in architecture. His idea was to capture individual solutions to common problems in his discipline and then re-apply and combine these solutions, with the ultimate goal to construct entire buildings by combining a sufficient number of pattern solutions. Alexander's idea found resonance not only within the domain of architecture, but also in other disciplines, most notably in informatics. Gamma et al. [1] provided the first comprehensive account of patterns as a tool to capture working solutions in software engineering and offered pattern requirements, a pattern classification, as well as a methodology to generate (or *mine*) patterns for reusable software engineering problem solutions.

The solutions described in a pattern are always specific to certain problems, which renders patterns less holistic than guidelines and other more general means of guidance. Some see this as a distinct advantage over guidelines [2], although it might be more reasonable to see guidelines as a supplementary – and not a competing – means to guidelines [3]. While patterns contain more specific and contextual information for solving a particular problem, they lack the general information and larger scope of guidelines. Even larger pattern collections are

only as comprehensive as the list(s) of problems they cover. A guideline can encompass a wide area with a clear beginning and an end, whereas a pattern collection can only ever cover a certain range of problems, which might or might not completely cover the respective area. Ontologies are another means of knowledge structuring and transfer with similarities to patterns [4]. An ontology is a tool to hierarchically structure concepts in a specific domain of knowledge [5]. Similar to patterns, they can be used to capture and promote the application of good or best practices [6]. Building such an ontology is considered to be a difficult task, even for experts [7] – a trait they share with patterns. Unlike with patterns, however, the reusability of ontologies is still considered a great and, as of yet, unsolved challenge [6,7], whereas reusability is one of the distinctive features of patterns [1].

Existing pattern methodologies and guidances are mostly specific to the discipline or domain they are generated for [8]. From a general point of view, patterns could be employed in a far greater number of disciplines, since a methodology to capture solutions to reoccurring problems should be of benefit for most, if not all, disciplines [9]. Despite this perceived benefit, available general literature on patterns is scarce [8], although not nonexistent.

Borchers [10] adapted the pattern approach to Interaction design and developed a model of patterns as interconnected entities, very similar to Alexander's original concepts of patterns in architecture. According to Borchers, patterns can be high- or low-level, depending on the problem's level of granularity and one high-level problem is often comprised of several low-level ones. This distinction is very similar to the one found in software engineering, with the difference that software engineering has a third category for the lowest-level patterns, the so-called *idioms*. Winn and Calder [11] Meszaros as well as Doble [12] developed pattern writing and structuring approaches and did so by developing and presenting their approaches as actual collections of patterns. While there have been considerable efforts by the Pattern Languages of Programs (PLoP) and related communities, general and discipline-nonspecific literature on patterns is more or less still in a state of relative infancy. The literature, which *is* available, often focuses on high-level guidance on pattern writing or theoretical concepts and frameworks common to all pattern approaches. Practical guidance, collections of lessons learned, or case studies on pattern generation with general relevance are difficult to find. That makes it difficult to apply the currently available body of general literature on patterns in a specific discipline – unless that discipline already uses patterns and has an appropriate body of specific literature available.

Ultimately, pattern mining is often a long and not very straightforward process. Beyond the theoretical guidance provided in the available literature, there are practical issues pertaining to the pattern mining process, which are beyond the scope of said literature. Such issues can be the question of how to choose the right individuals to participate in each stage of a pattern mining process, how solutions should be presented visually in a respective community (technical schematics vs. supporting illustrations vs. plain text), questions of copyright of working solutions relative to the legislatory regulations in a specific geographical

area, a.s.o. For our pattern approach, we had to look for strategies to deal with these and similar issues. In the next section, we provide brief overview of our approach before we begin listing and discussing the actual issues.

3 The HCI Automotive Pattern Approach - An Overview

In the Christian Doppler Laboratory for "Contextual Interfaces we are collecting pattern solutions for recurring HCI design problems in the driver space domain[1]. It is a collaborative effort together with our technical partner AUDIO MOBIL, Ranshofen, Austria. In the project, we have decided to pursue a nonstandard pattern mining approach by drawing from both academic and industry resources. This is due to the exploratory nature of the research, in which we look for solutions to often novel problems with very new technologies, to which established solutions do not yet exist (either at all or at a large scale). That does not mean, however, that proven solutions to *parts* of such problems do not exist as well. That is why we decided to explore partial solutions from both angles (academia and industry) in order to arrive at pattern solutions for the actual, larger problems.

Our pattern generation[2] process begins with an initial problem finding and knowledge transfer workshop. In this workshop, experts from both academia and industry are present. The concept and methodology of patterns is explained to everyone in order to ensure common ground. Then, a focused brainstorming for domain-specific problems is made, building the basis for the initial pattern mining. This initial mining is done by the HCI researchers, in which they break down the overall problems into smaller sub-problems and then look for state-of-the-art solutions in the automotive industry, as well as lab-studies or prototypes in academic literature which can solve the problems. These solutions are collected and written down as a pattern afterwards, based on a predefined pattern structure [3].

These initial patterns are then discussed and rated in a workshop, in which both HCI researchers and industry stakeholders participate. Each pattern is rated in every pattern category on a 5-point Likert scale via a standardized rating sheet [14] and discussed separately. After the workshop, the second iteration cycle begins. Each pattern is iterated by at least one industry stakeholder and one HCI expert. Once all patterns are iterated, a second workshop is conducted, which proceeds exactly like the first one. The patterns are then reworked once more and validated via the rating sheet one final time. If they pass in each category (higher than 3), they are considered final; if they do not, they are reworked again and integrated into the appropriate evaluation workshop, once the next new batch of patterns is being generated. While this overall process description is, while brief, rather complete, there was a number of smaller issues that appeared during individual stages of our pattern activities, which we discuss in the following sections.

[1] Examples for such patterns can be found in [13].

[2] We use 'pattern generation' for referring to the overall process, encompassing problem finding, pattern mining and pattern writing.

4 Lessons Learned

In this section, we present five problem areas we encountered during our pattern generation process and some lessons learned and strategies we used to solve them. The pattern context described in this paper is a rather particular one, both due to the domain (HCI) and procedure (academia and industry as sources). Nevertheless, the issues and lessons learned are very practical in nature and should be relevant to other fields and disciplines, which employ patterns as well.

4.1 A Matter of Language

Due to the interdisciplinary approach we pursued, the language used in our patterns is strongly based on scientific papers and books. It is easier for researcher to transfer the knowledge from referenced works to a solution, when maintaining the characteristic precision of scientific language, together with the writing and argumentation styles of the respective discipline. Writing patterns in a language, which is common to most individuals and easy to understand is often a basic requirement of a successful pattern. Industry stakeholders are not always researchers and when they are, they do not always use the same terminology and writing styles as scientists do. Thus, even though the domains (Human-Computer-Interaction and Driver Space Design) were similar, the scientific language we used in our initial patterns (e.g., UX factors, acronyms, or statistical study descriptions) turned out to be difficult to understand for the industry stakeholders.

Therefore, simplification of the used language is a must, in order to provide easy to read solutions. However, it is difficult to define which language is complicated and which is easier to understand. This heavily depends on the language skill level of the reader in question. Finding the right language is a balancing act. Patterns need a language to offer simple solutions with exact values without oversimplification and information loss. But simplification alone, as it turned out, is not everything. In our second pattern iteration, we had removed as much scientific jargon as possible, employed natural instead of specific writing styles, and added explanations for concepts, which might be obvious in the scientific community but not everywhere else. However, the stakeholders still expressed dissatisfaction with wording, sentence structure and terminology. We found that stakeholders held certain expectations, which they were unable to explicitly state, due to those expectations being dependent on technical and other literature they read on a daily basis (e.g., technical manuals, schematics, patents, but also technical news articles). Furthermore, in their often very competitive environment, there is no guarantee that stakeholders will admit their lack of knowledge or language skill if e.g. certain definitions mentioned in solutions are unclear or unknown.

All these aspects were making it hard to adjust the language used in the patterns. We could not simply ask the stakeholders about how they would like the pattern to be written. But it would also not have been feasible to go through all their everyday literature and develop a suitable writing style from that. The

solution we eventually arrived at was rather simple: We provided one pattern, which had received high rating in most categories other than language and had each stakeholder rewrite the pattern according to their needs. We collected and consolidated these rewritten patterns and used this as a writing template for all future patterns. This made it easier to adjust language use, level of detail in explanations, etc. for future patterns, which received better ratings regarding language used and comprehensibility.

4.2 Cross-Discipline Solution Searching and Unreliable Sources

The problems we tried to solve in our pattern collection were often focusing on very specific topics, which are not well researched in a given domain. In other cases, they covered multiple research areas. In order to adequately cover these areas, we drew from research articles, books, guidelines, standards (e.g., ISO, DIN), industry state of the art approaches, or prototype designs and demos from different areas for finding our solutions. This lead to two issues. Firstly, the individual sources we drew from did not follow the same purpose in most cases, which meant that it was difficult to consolidate them all into single solutions. Secondly, in tying information from different sources together, we encountered two specific problems regarding materials covering very narrow areas, as well as sparsely documented implementations and prototypes.

Generally speaking, if the underlying problem is focusing on a very specific topic and no research is to be found, the research area has to be expanded. Expanding the investigation area by including research from different but related fields can provide valuable content needed for solution finding. Of course, this is more easily said than done, since it is not always immediately obvious, which domains are suitable candidates for such an expansion. We decided to pursue a simple pragmatical solution in which we extracted key aspects from the problem description and transforming these into a set of *problem keywords* (e.g., "eye movement, position, output, visual" was one such set for a pattern, which tried to identify ideal display positions depending on eye movement speeds). With the help of a web based search engine, preferably using meta data of scholarly literature, these words are then used as search terms. Commonly this method requires several trials, using different combination of the problem keywords while trying to leave out the least significant ones, before leading to adequate results. Of course, these results still had to be checked individually for their actual relevance, but the method at least provided sources to work with and usually produced a number of relevant and usable sources after some time of searching and sorting out the irrelevant ones.

When dealing with topics like state of the art approaches and prototypes, there was usually very little research or supplementary material available. Thus, we had to look for information sources other than academic research. Those sources were often news articles, on-line videos, or presentations. These sources are less strictly evaluated than academic works and should be handled with appropriate caution. Before using sources like these as references for solutions and examples, it is strongly advised to review and verify the content.

Videos, presentations as well as news articles might offer some useful information for state of the art approaches, new prototypes or future technologies, but are unlikely to provide exact values. Even though the use of videos may be somewhat limited for exact solutions, they proved to be very handy for the example section of our pattern approach. The quite simply take-home message we took from this was "It's better than nothing", as patterns with supplementary information, even when vague, where perceived better than patterns without that information.

4.3 Of Iterations and Different Points of View

Workshops with industry stakeholders in the iteration process are a good opportunity to discuss the pattern work progress. The iteration sessions as well as the pattern rating workshops are generally organized after a certain pattern work package (usually anywhere between 5 to 12 patterns) is accomplished.

Internal iteration processes with HCI researchers are held more frequently and are used to improve the overall quality of the patterns. Iterations are aimed at reviewing the most recent pattern changes and discuss possible further issues. These issues are usually rather minor and can reach from changing the title of a pattern to be more precise, to clarification of problem topics or solutions for the sake of comprehensibility. New patterns and patterns being iterated for the first time, are inspected more thoroughly. However, during internal iterations the focus is mostly on clarity, comprehensibility, and structure. Issues regarding the pattern content are mostly left out at this stage. Questions and uncertainties, emerging during pattern creation and internal iteration, which are not answerable by the HCI researchers at this point, are noted down for subsequent external iterations and workshops with the industry stakeholders.

During workshops the generated patterns are reviewed and evaluated by the industry stakeholders. The industry stakeholders are usually the source of information needed to finalize the patterns. This happens in a two-step process. The industry stakeholders are given each pattern and enough time to read through it thoroughly. After that, they are instructed to rate each pattern as described in Sect. 3. During this rating stage, the stakeholders are additionally instructed to note down particular issues they would like to raise later on. These can be related to one individual pattern or the whole pattern collection. After a certain batch of patterns has been rated, the second step begins and a discussion is held. In this discussion, the stakeholders supplement their ratings with qualitative input. On top of identifying individual iteration needs, this also helps the researchers understand the industry stakeholder's way of thinking, which plays a crucial part in providing the solutions in an adequate manner. Pattern solutions have to solve the problem statements and provide exact values in a form or unit the industry stakeholders need in order to implement the solutions. These iteration workshops are, therefore, the ideal opportunity to not only handle the content-side of the pattern iteration process, but also to define a set of preferred metrics and units for pattern solutions, depending on aspects such as modality or design

target (e.g., *db* for acoustics, *lm* for lighting, distance from center to center in *mm* for arrangement of input buttons, etc.).

4.4 Linking - Online vs. Paper

Solution examples provided in our patterns are mostly text based, but supplemented with images, tables, hyperlinks, as well as videos. For the most part there are few to no limitations in accessing supplementary information in printed media. That is not the case for hyperlinks and videos, however. In order to access online content like videos, databases or similar content, hyperlinking is necessary. Handling those proved to be more of a problem than we had initially expected. While it is possible to provide hyperlinks in form of written text, accessing them with ease is only possible in digital form. Links contained in physical print media need to be typed into the browser's address bar manually in order to be accessed. Furthermore, a person who uses a printed pattern collection might not always be in the vicinity of a workstation with internet access. Ideally, pattern solutions should be quickly accessible, so this is an obvious limitation. This limitation reduces the options of how we can provide solution examples in our patterns in printed form. In the first versions of our patterns, we tried to handle hyperlinks to videos as simple as possible. We provided an image showing a scene of the video, followed by a link, which could be clicked on in the digital version of the pattern. While it was possible to get a first impression of what would be shown in the video through the provided image, the link still had to be written down manually in the printed version (Fig. 1).

Fig. 1. Link to https://youtube.com

The preview image could not fully represent the video contents, so our solution was still sub-optimal and in need of improvement. To overcome the limitations of our initial versions, we decided to use Quick Response codes in addition to the regular hyperlinks. QR Codes[3] are a type of two-dimensional bar-code, which can be scanned by an imaging device. Scanning the code with a smartphone converts the QR Code into a useful form e.g. a hyperlink to a webpage,

[3] Short term for Quick Response Code.

database or video, which effectively eliminates the need to write down the link manually. Thus we provide solution examples for e.g. videos by adding QR-Codes containing a link to the videos instead of a text only link (Fig. 2). In addition to the QR Code, the link itself in text form is still available as a reference.

Fig. 2. QR code for the link to https://youtube.com

This solution was much better received by our pattern users although it does come with two drawbacks. For one, each QR code takes up more space than a hyperlink, which is usually just a single line of text. Secondly, they require access to a device with QR reading capability. Most smartphones of today do have this capability, however, so this can be considered a rather minor limitation. Considering the benefits and drawbacks, it is an improvement over the previous versions and we decided to adopt QR codes for all our patterns, as it lead to better usable patterns overall.

4.5 The Art of Referencing

The amount of sufficient references needed to provide enough information (not appointed in the method) depends on the scope the pattern is trying to fill. References we used in our patterns ranged from research articles over books and guidelines up to videos and prototypes. Each of these information sources had to be cited in out patterns adequately. We initially thought this to be a rather trivial issue and simply applied the same standards one would apply for any scientific work: if it is present in another source, then it needs to be cited. However, it turned out during initial iterations with industry stakeholders, that actually following this quite basic principle caused quite a few problems. It was perceived as very irritating to have the text broken up by references and citations every few lines, which was compounded by the fact that we had used the author-year format for the initial versions of our papers. Thus, we had to find a way to (a) reduce the space taken up by each individual reference, as well as (b) the overall amount of references, in order to render the patterns more readable for the industry stakeholders.

The first part was rather simple – we switched from the author-year to the number format, which took up less space and was immediately deemed acceptable by the stakeholders during subsequent iterations. Dealing with the question of the right amount of citations was an entirely different and more difficult matter, however. Even in our adapted and slightly atypical pattern approach, the

solutions should still be proven ones. Tying a solution (or parts of it) to multiple references is one way of satisfying this provenness-criterion. In addition, references not immediately relevant to an individual problem might tie into other related problems or a larger overall problems. Still, we tried to minimize the amount of references *required* for a successful and more easily readable pattern by establishing and comply the following simple rules for choosing references to cite in the pattern:

- References with limited to no use for a certain problem are not mentioned. This includes references to other related problems or larger problems the individual problem is a part of. Such references should be handled via keywords within the pattern collection.
- The aim of the HCI patterns is to provide solutions with exact implementation metrics. Thus, each reference cited must either provide concrete values or a full solution implementation description.
- The use of only one reference is legitimate, when it provides exact values, thus a perfect solution for a certain problem topic. Note that this rule is tiptoeing the provenness-condition.
- There is more than one way to solve most problems. Multiple references with exact values should not be inconsistent with each other or the provided solution. If exact values are provided by multiple references, they should be consistent with each other
- A solution is backed up with sufficient references, if the categories *solution description pattern completeness* receive a sufficiently high rating (i.e., higher than 3 in our rating system).

Following these rules has worked rather well so far but it needs to be said that the pattern approach is still a work in progress and these rules for citation are not yet final. It should also be noted that a reduction of sources can lead to better readability at the cost of validity and profoundness of the solution description, so it might well be that there is a better solution, which allows to make a high volume of references easily digestible by a non-academic readership. We have, however, not found such a solution at this point.

5 Conclusion

In this paper, we presented and discussed some problems we faced during our collaborative and interdisciplinary pattern generation activities. We then provided the solutions we developed for each of these five problems and briefly discussed these. In the following, we give a quick final summary of each of our solutions as a lesson learned for future pattern generation activities.

- **Lesson One:** Matching the language of pattern writer and reader is a difficult task, especially since the prospective pattern readers are not always able to fully express their preferred language and writing style. Having the pattern readers do one pattern iteration of a content-complete pattern regarding only

writing, typography, terminology, and argumentation, is a relatively quick and easy way to get a template for matching languages in pattern writing.
- **Lesson Two:** When one discipline does not cover a topic to a sufficient degree or implementations are not documented or supplemented well enough, one needs to search "outside the box" for solutions. Breaking down the problem into a keyword structure and using those in a meta data search provided to be fruitful for cross-discipline solution searching. Unstable and unreliable sources such as news articles, on-line videos, and presentation slides often help to underpin sparsely documented solution implementations. They might not be validated and should be treated with caution, but they provide a background to the solution and foster understandability of the pattern more than the complete absence of any supplementary material would.
- **Lesson Three:** Academic's and industry stakeholder's needs do not always align. Internal and external iteration processes should be aligned such to foster this alignment. Polishing regarding structuring and formulations should be handled in internal iterations as much as possible, so that the external iterations can be used for content-related issues. External iterations and exchange should additionally be used to define concrete values and units for the solution descriptions, based on modalities and design space targets.
- **Lesson Four:** Linking to online resources and videos is difficult in printed pattern volumes, as typing out the URLs can be tedious and time consuming. A preview image of the video is not enough to fully communicate the video content, so a better solution is required. QR codes provide easier access to online resources at the cost of space (=longer patterns), which is an acceptable tradeoff for more usable and readable pattern.
- **Lesson Five:** References are less important for industry stakeholders than they are for academics. When choosing which works to reference, one should focus on sources, which provide concrete values or full solution implementation descriptions. When multiple solutions are valid, only one should be chosen and described. If a pattern is rated as complete and the solution as adequately described, then it contains sufficient references. To put it in a nutshell: less is more.

The problems and the solutions we provide are very practice oriented and are based on individual feedback, pattern rating results and discussion sessions during numerous pattern iteration workshops. The lessons learned should not be considered universally valid. However, they worked in our particular case to provide higher quality and better readable patterns. There is certainly room for improvement for many, if not all, of the solutions provided, which we will do in our continuous pattern approach development. We encourage others working with patterns to do the same and to develop and share knowledge regarding practical issues of pattern mining and writing, in order to further knowledge on pattern generation within and across communities and disciplines.

Acknowledgments. The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development and AUDIO MOBIL Elektronik GmbH is gratefully acknowledged (Christian Doppler Laboratory for "Contextual Interfaces").

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education, Upper Saddle River (1994)
2. Van Welie, M., Van Der Veer, G.C.: Pattern languages in interaction design: structure and organization. In: Rauterberg, M., et al. (eds.) Proceedings of Interact 2003, pp. 527–534. IOS Press (2003)
3. Mirnig, A.G., Meschtscherjakov, A., Perterer, N., Krischkowsky, A., Wurhofer, D., Beck, E., Laminger, A., Tscheligi, M.: Gaining user experience patterns by drawing from science and industry: a combinatory pattern approach. Int. J. Adv. Life Sci. **7**(3–4), 145–157 (2015)
4. Gangemi, A., Presutti, V., Blomqvist, E.: The computational ontology perspective: design patterns for web ontologies. In: Sartor, G., Casanovas, P., Biasiotti, M., Fernández-Barrera, M. (eds.) Approaches to Legal Ontologies. Law, Governance and Technology Series, vol. 1, pp. 201–217. Springer, Cham (2011). https://doi.org/10.1007/978-94-007-0120-5_12
5. Blomqvist, E., Sandkuhl, K.: Patterns in ontology engineering: classification of ontology patterns. In: ICEIS, vol. 3, pp. 413–416 (2005)
6. Falbo, R.A., Guizzardi, G., Gangemi, A., Presutti, V.: Ontology patterns: clarifying concepts and terminology. In: Proceedings of the 4th International Conference on Ontology and Semantic Web Patterns, WOP 2013, Aachen, Germany, vol. 1188, pp. 14–26. CEUR-WS.org (2013)
7. Poveda-villalón, M., Suárez-figueroa, M.C., Gómez-pérez, A.: Patterns in ontology engineering: classification of ontology patterns. In: Proceedings of Second Workshop on Ontology Patterns (WOP 2010), pp. 35–49. CEUR-WS (2010)
8. Mirnig, A.G., Tscheligi, M.: Introducing a general multi-purpose pattern framework: towards a universal pattern approach. Int. J. Adv. Intell. Syst. **8**(1–2), 40–56 (2015)
9. Vlissides, J.: Pattern Hatching: Design Patterns Applied. The Software Patterns Series. Addison-Wesley, New York (1998)
10. Borchers, J.O.: A pattern approach to interaction design. AI & Soc. **15**(4), 359–376 (2001)
11. Winn, T., Calder, P.: A pattern language for pattern language structure. In: Proceedings of the 2002 Conference on Pattern Languages of Programs, CRPIT 2002, Darlinghurst, Australia, vol. 13, pp. 45–58. Australian Computer Society, Inc., Sydney (2003)
12. Meszaros, G., Doble, J.: Pattern Languages of Program Design 3, pp. 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston (1997)
13. Kaiser, T., Mirnig, A., Perterer, N., Meschtscherjakov, A., Tscheligi, M.: Car user experience patterns: a pattern collection in progress. In: Proceedings of the 8th International Conferences on Pervasive Patterns and Applications, PATTERNS, pp. 9–16. IARIA (2016)
14. Wurhofer, D., Obrist, M., Beck, E., Tscheligi, M.: A quality criteria framework for pattern validation. Int. J. Adv. Softw. **3**(1–2), 252–264 (2010)

Interactive Dashboard for Workflow Engine Benchmarks

David Bimamisa, Mathias Müller, Simon Harrer(✉), and Guido Wirtz

Distributed Systems Group, University of Bamberg, Bamberg, Germany
{simon.harrer,guido.wirtz}@uni-bamberg.de

Abstract. Today, more and more companies model their business processes using languages such as Business Process Model and Notation (BPMN) or Business Process Execution Language (BPEL), and automate their processes by executing these models on appropriate workflow engines. To help choose the best fitting engine among different alternatives, several benchmarking initiatives have emerged, e.g., the BPEL/BPMN Engine Test System (betsy) that benchmarks conformance and expressiveness, and BenchFlow that benchmarks performance. However, their results are hard to analyze and compare for the typical end-user, developer or researcher. This paper tries to solve this issue by introducing (a) a common data model which can hold data of both betsy and BenchFlow tests and their results, (b) a transformer that can automatically transform the results of betsy and BenchFlow runs into the common data model, and (c) an interactive dashboard that visualizes the results according to the most important use cases. Hence, it enables the end-users, developers and researchers to analyze and compare the engines in a straight-forward manner.

Keywords: BPMN · BPEL · Interactive dashboard · Benchmarking Workflow engine

1 Introduction

Over the last few years, several process languages (e.g., BPMN [13], and BPEL [14]) have been developed and standardized for modeling business processes and for building service-oriented and process-aware systems. Today, many business process execution engines (or Workflow Management Systems (WfMSs)) exist that support the execution of process models. A workflow is the automated part of a business process [20, p. 8], and if the process is already fully automated they are used synonymous. But with the emergence of workflow engines, choosing the "best-fitting" [5] workflow engine for a project becomes more challenging [3]. Typically, end-users and companies aim to choose a workflow engine based upon quality characteristics such as usability, performance and language support [5]. However, evaluating and comparing workflow engines with respect to the desired characteristics is a complex undertaking.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 176–188, 2018.
https://doi.org/10.1007/978-3-319-72125-5_14

Benchmarks are often used by end-users to evaluate and compare competing systems before deciding on the system to adopt [15]. Likewise, researchers and developers use benchmarking techniques to evaluate the design and improvement of systems over time. In the field of workflow engines, benchmarks for testing different capabilities of workflow engines have been built in [3,6,8]. These benchmarks may support users, developers and researchers in analyzing and comparing workflow engines for BPMN and BPEL. While [6,8] have conducted benchmarks for assessing the standard conformance and expressiveness of BPMN [6] and BPEL [8], Ferme et al. [3] have evaluated the performance of workflow engines for BPMN. These papers only show aggregated data of one or two engine capabilities to get their point across. The actual raw data is sometimes published as well, but uses different data formats and levels of detail. Hence, working with this raw data is either not possible at all or very hard. But especially the raw data would allow to gain additional insights, e.g., a clear picture of different capabilities per engine. As with many benchmarks (e.g., for databases, Web servers), both benchmarks for workflow engines only show raw data with few aggregation and less emphasis on visualization and interaction with the data. While raw data and non-interactive result tables are sufficient and even preferred [21, p. 52] by expert users, novice end-users may find it difficult and time-consuming to choose their best-fitting workflow engine with this level of detail. In contrast to experts, novices may have not enough knowledge to analyze, compare and identify problems using atomic raw data. Also, raw data may not meet the different cognitive skills (e.g. analytical versus less analytical) [21, p. 49] of end-users. It has been shown that an appropriate aggregation and visualization of data helps users in understanding complex information, identifying problems and focusing on the most relevant data (e.g. metrics) for better decision making [21]. Furthermore, at the moment, benchmark data for workflow engines are not publicly available or accessible. This makes it difficult for end-users to make informed decisions about choosing an appropriate workflow engine without benchmarking these engines themselves.

To overcome the aforementioned limitations, we created a web-based interactive dashboard that makes workflow engine benchmarks accessible, visible and comprehensible to end-users. By building a dashboard, we mainly focus on empowering non-expert end-users to easily understand, analyze and access benchmark results. Nevertheless, experts may still use the dashboard to manage their own benchmarks, to compare them with our results or to update test implementations. This could establish a community of discussing and sharing engine benchmarks. Further, when using a dashboard, experts may profit from the more aggregated and structured presentation of the benchmark results to effectively communicate them to non-experts stakeholders [16].

In this paper, we present the design and implementation of the interactive dashboard for comparing workflow engines. It visualizes the benchmarks of the benchmarking tools betsy and BenchFlow [3,6,8], but it is designed to support other engine benchmarks as well. First, we outline related work along with the used terminology in Sect. 2, followed by the listing of requirements the interactive

dashboard shall fulfill in Sect. 3. Next, in Sect. 4, we describe how we aim to implement the requirements, namely, designing a common data model (see Sect. 5) and then implementing the interactive dashboard and a transformer mapping the data models of betsy and BenchFlow to our data model in Sect. 6. Finally, we conclude this paper with an outlook on future work in Sect. 7.

2 Related Work and Background

2.1 Dashboards

Few [4] defined a dashboard as "a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance". Based on this definition, we define a dashboard for software benchmarks as a tool to collect, summarize and present a large set of benchmark results from multiple sources over time in single web pages so that key performance indicators of benchmarks can easily be perceived and understood at a glance [21]. In relation to this work, a web-based dashboard for benchmarks consists of web pages displaying most important information about the results of different benchmarks (e.g., conformance, expressiveness, performance). While dashboards are commonly used as part of business intelligence (BI) systems for measuring and monitoring business performance to support managerial decision making [21], their characteristics and design principles have been adopted for reporting benchmark results.

In the following we highlight common characteristics and advantages of web-based dashboards for visualizing benchmark results. Benchmark dashboards typically collect and summarize data from different benchmark runs to allow the analysis and the comparison of these benchmark runs. Some dashboards collect benchmark results of the same capability over a period, which may be helpful for identifying trends such as the stability or the improvement of the tested system. A web-based dashboard is a simple method for making a large set of benchmark data available in one place on the web, and thus easy to find, access and share. When done right, dashboards display few metrics and aggregated information to reduce the information load, and to help users quickly get an overview of the benchmark results. Interactive dashboards presenting aggregate data mostly allow users to navigate from highly aggregated data to more detailed level of data, e.g., to obtain additional details on a particular benchmark run. This feature is known as drill-down [16]. Furthermore, interactive dashboards often provide mechanism for filtering data to quickly find and compare specific parts of the benchmark results. More importantly, filters allow the exploration of data which is useful for users unfamiliar with the data [2]. One important characteristic of dashboards is the focus on making the visual presentation of data intuitive and easy to understand. This is typically achieved by making use of colors, distinctive sign (e.g., +, −), charts and graphs, easing the identification of key metrics and the comparisons of the results.

2.2 Dashboards for Benchmarks

The characteristics and design principles of dashboards have been adopted in a variety of web-based dashboards for reporting benchmark results. For example, TechEmpower Framework Benchmarks[1] (TFB) is a dashboard that presents performance benchmark results of various web frameworks to facilitate the comparison of these frameworks regarding their performance. TFB has become a community-driven project in which contributors frequently update test implementations. Compat-table[2] is a dashboard that presents benchmark results that evaluate the compatibility of browsers, servers and compilers with ECMAScript 5, 6 and 7, while the Node-compat-table[3] dashboard only focuses on evaluating different versions of the Node.js server regarding ECMAScript support. Both dashboards make extensive use of colors to simplify the differentiation and the recognition of the result ratings. Also, they employ drill-down features to enable the aggregation of the ECMAScript features. JMeter[4] is a performance testing framework that comes with a dashboard for displaying the benchmark results using tables and graphs. It provides filters to specify which rows of tables or series of graphs should be shown or hidden. Another benchmark framework that provides a dashboard is Rally[5]. It is used for benchmarking the OpenStack[6] cloud hosting infrastructure. The Rally dashboard simplifies the collection of test results and extracts relevant metrics to help users improving their OpenStack infrastructure. Similar to our dashboard, it also provides information about the configuration of each test (e.g., test cases, configurations).

Despite the existence of various web dashboards for benchmarking tools, there is no *interactive* dashboard yet for presenting the benchmarking results of workflow engines. To the best of our knowledge, only the presentation of the workflow patterns[7] has few characteristics of a dashboard, but it displays only result tables with no interactivity.

2.3 Workflow Engine Benchmarking

Within the area of workflow engine benchmarking, the two most prominent frameworks are the BPEL/BPMN Engine Test System (betsy) and BenchFlow. Betsy can benchmark various capabilities, including conformance [6,8], static analysis conformance [11], expressiveness [7,9], and robustness [10] of a plethora of BPEL and BPMN engines [5]. BenchFlow is an EU project with the aim to develop a performance benchmark for workflow engines. It is able to benchmark three BPMN engines for performance related metrics of, e.g., workflow patterns [3,18].

[1] https://www.techempower.com/benchmarks.
[2] http://kangax.github.io/compat-table.
[3] http://node.green.
[4] http://jmeter.apache.org/usermanual/generating-dashboard.html.
[5] https://www.mirantis.com/blog/rally-as-an-openstack-performance-dashboard/.
[6] https://www.openstack.org/.
[7] http://www.workflowpatterns.com/.

Next to betsy and BenchFlow, there are other benchmarking approaches regarding workflow engines. [1] compares engines supporting BPMN, BPEL and even XPDL, but their approach is not fully automated and their evaluation is rather high level. Nevertheless, their results could be incorporated into this dashboard as well. The results of nine BPEL performance benchmarking approaches [17] could also be integrated.

2.4 Terminology

The relevant terms used in this paper and that are at the heart of the dashboard are defined as follows:

- **Engine instance:** An engine with a specific version and an optional configuration. For instance, `activiti 5.15.1` or `Apache ODE 1.3.6 in-memory`. An engine normally has multiple engine instances.
- **Process:** A process is defined using a process language (e.g., BPEL or BPMN). As we only talk about executable processes, the term workflow is synonymous in this context.
- **Capability:** A capability is either conformance, expressiveness or performance in this context. We understand that a capability is a quality characteristic that can be associated within one of the characteristics of the ISO/IEC 25010 standard [12].
- **Conformance Benchmarking:** Evaluates which (configuration of a) language construct is supported by the given engine instance.
- **Expressiveness Benchmarking:** Evaluates the degree of support of workflow patterns of a given engine instance in relation to the degree of support of the process language [9,19].
- **Performance Benchmarking:** Evaluates the performance and resource utilization of a process on a given engine instance using a specific workload [3].
- **Test:** Evaluates a single testable feature of a capability using a specific process on a given engine instance.

Each capability defines its own set of testable features and the aggregation hierarchy of feature sets and groups as shown in Table 1. Regarding conformance, a construct is a building block of the process language (e.g., `ExclusiveGateway` of BPMN). Of each construct, we derive testable features by instantiating the construct with a specific configuration, for instance, an `ExclusiveGateway` with

Table 1. The different feature trees per capability

Capability	Conformance	Expressiveness	Performance
Group	Construct group	Pattern catalog	-
Feature set	Construct	Pattern	-
Testable feature	Construct config.	Pattern impl.	Workflow

a default `SequenceFlow` in BPMN. A construct group is simply a set of semantically related constructs, e.g., all `gateways` in BPMN. Regarding expressiveness, we have pattern catalogs that consist of patterns which can have different pattern implementations. The pattern implementations are the testable features. For instance, we have the `workflow control flow patterns` as a pattern catalog, with the pattern `WCP-01 Sequence` which has one pattern implementation. Regarding performance, the testable features are one or more workflows which are put under test by applying a specific workload within a specific environment. As a result, the performance and resource utilization is measured. But as performance metrics are dependent on the actual hardware and software resources of the system on which the testing was conducted, an aggregation does not make sense.

3 Requirements

This section discusses the requirements for the dashboard. We can distinguish between two different stakeholders, namely, the *user* representing anyone who either wants to use or already uses a workflow engine, and the *vendor* representing anyone who develops, sells or provides support for an engine. The user is looking for a workflow engine to use and wants to know which one fulfills his needs best, and which one would lead to the least danger of a vendor lock-in. But it could also be the case that the user already uses a workflow engine and is thinking about migrating to another one and wants to know which engine supports the most of the required features. A special case would be the migration from an old(er) version of a workflow engine to a newer one. In contrast, the vendor is looking forward to know how its product compares to its competitors and how their own product has evolved over time featurewise.

The requirements were elicited by analyzing the benchmark results of betsy and BenchFlow, by questioning experts from different universities and by reviewing scientific papers and existing web dashboards. The captured requirements were decomposed into user stories and validated by letting the experts try out an interactive prototype. The most important requirements for the dashboard are:

R.User.1 – compare latest feature: As a user, **I want to** see differences and similarities in feature support of the latest engine instances **so that I can** make an informed decision which engine to choose.

R.User.2 – compare latest aggregated: As a user, **I want to** see differences and similarities in feature support on an aggregation level of the latest engine instances **so that I can** make an informed decision which engine to choose more quickly.

R.User.3 – compare versions feature: As a user, **I want to** see differences and similarities in feature support of multiple engine instances of the same engine **so that I can** check if an upgrade is worthwhile.

R.User.4 – drill down to feature: As a user, **I want to** see all features for each (unsuccessful) aggregation level **so that I can** check if all features are

unsupported or if there are only problems with some of the features in this aggregation.

R.User.5 – compare any feature: As a user, **I want to** see differences and similarities in feature support among different engine instances **so that I can** estimate the migration costs between engines (see vendor lock-in).

R.User.6 – compare all aggregated: As a user, **I want to** see aggregated information of all engine instances in one place **so that I can** quickly gain an overview of all engines as an entry point for further investigations.

R.Vendor.1 – compare latest: As a vendor, **I want to** compare the strengths and weaknesses of the latest engine instances **so that I can** know how my engine compares to its competitors.

R.Vendor.2 – compare engine progress: As a vendor, **I want to** compare multiple versions of the same engine **so that I can** reveal the feature progress of the engine.

R.Vendor.3 – detail single engine: As a vendor, **I want to** evaluate the strengths and weaknesses of a single engine instance **so that I can** see how it can be improved further.

R.Vendor.4 – drill down to failure: As a vendor, **I want to** be able to drill down to the cause of any failed feature on my engine **so that I can** verify and fix the failure.

In addition to these functional requirements, two additional constraints were imposed to the dashboard, namely, that it should not use any server-side logic to keep hosting simple and secure, and that it is easy to use and visually appealing.

4 Big Picture

The big picture of our approach to fulfill the requirements in Sect. 3 can be seen in Fig. 1. It contains three data models and four software components. Both betsy and BenchFlow already have their own workflow benchmark model in which they describe their benchmarks along with their results. The dashboard itself requires a common data model which contains information of both betsy and BenchFlow. The transformer has the responsibility to map the data from

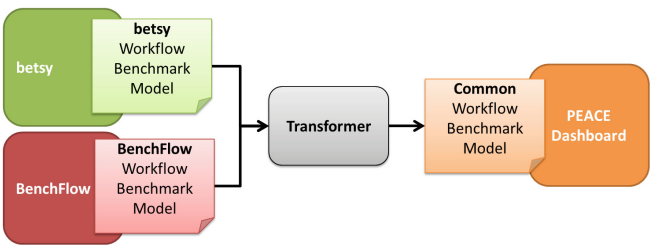


Fig. 1. Big picture

the two workflow benchmark models of both betsy and BenchFlow to the common workflow benchmark model for the dashboard. Moreover, it can enrich the data or apply integrity checks during the mapping as well. For example, the transformer adds images of the workflow models that are tested to the data and checks if all referenced files exist. It also checks the data for uniqueness and existence of referenced IDs. The end user only sees the dashboard.

5 Common Data Model

We created a common data model based on the data models of both betsy and BenchFlow. In the **betsy model**[8], a test evaluates whether a BPMN or BPEL process behaves as expected. It comprises test cases with test steps that inject input into the workflows and associated assertions which evaluate the state of the workflows. A feature is considered to be supported if all test cases pass. If only some test cases pass, the feature is partly supported, and if none pass, the feature is not supported. If a workflow itself is undeployable, it is not supported as well.

The **BenchFlow model**[9] is structured into trials and experiments. In a trial, one or more BPMN process models are deployed to the engine under test and tested using an given load function (i.e., workload) while measuring various metrics (e.g. cpu and memory usage, throughput in form of number of executed workflow instances per second and duration/number of the workflow instance). Trials which test the same process model using the same workload on the same environment are aggregated into experiments. In [18], each experiment comprises three trials, and is conducted on three engines using various workloads.

To sum up the differences of both data models, betsy tests features in isolation and quantifies the result into something meaningful, whereas BenchFlow repeats its feature tests multiple times (trials) and aggregates these results to reduce errors of measurement and to make the data meaningful. Moreover, betsy does not provide any additional data besides the test itself, whereas BenchFlow gathers information about the environment and the underlying hardware and software system as well.

Based on the *betsy model* and the *BenchFlow model*, we created a *common data model* that represents both types of benchmarks. Figure 2 shows a high level version of this model alongside a shortened example of a single test. The common data model consists of four main elements: the engine, the benchmarking framework, the feature (in its feature tree) and the test. The benchmarking framework is identified by its name and version. In contrast, an engine is identified by its name, version and configuration parameters. Hence, an engine with a different configuration is another engine in our terminology. Furthermore, for

[8] https://github.com/uniba-dsg/betsy/tree/master/src/main/groovy/betsy/common/model.

[9] https://github.com/benchflow/docker-images/blob/dev/cassandra/data/benchflow.cql.

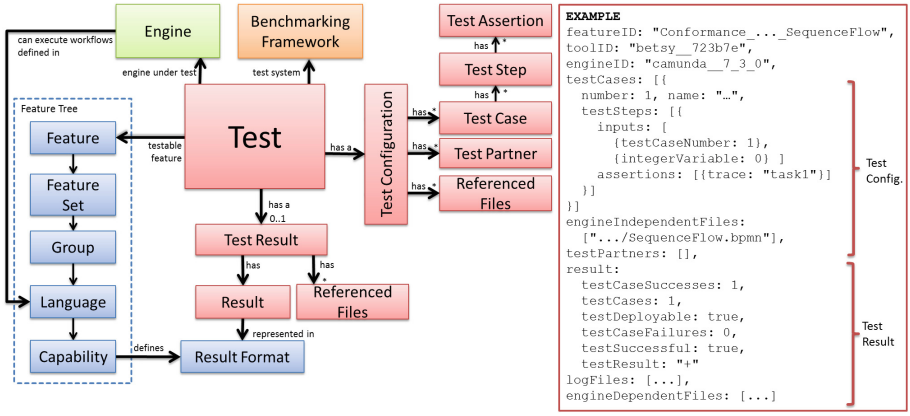


Fig. 2. The common data model

each engine the release date, its URL, software license[10], and the programming language it is developed in is stored. This data reveals internals of the engine which can help end users when using and developers when extending the engine. The ID of the feature consists of the names of all elements along the tree, starting from the capability, language, group, feature set, and feature. Hence, the feature tree is hierarchically structured and provides two levels to categorize *features* with the *feature sets* and *groups*. Features, feature sets and groups have different meanings depending on the tested capability as shown in Table 1.

The test is the key element in the common data model. It tests a single feature of an engine using a specific benchmarking tool, and because of this, its ID is the triple of the IDs of the feature, the benchmarking tool and the engine. Each test has a name and a description. The test configuration of a test is also known as the engine independent part of the test and consists of referenced files (e.g., the process models, required XML schema files, etc.), the test cases with their steps and assertions, and optional test partners so that the workflow under test can interact with third party services. The test results, however, are engine dependent, and comprise various referenced files such as logs or the deployed workflow model, and the actual results which are in a result format that is dependent on the capability the tested feature corresponds to. In addition, both the test results and the test configuration can have additional data stored in a key-value based tree to handle any additional tool specific data, which is not displayed in Fig. 2.

The test results are dependent on the tested capability. For conformance and expressiveness benchmarking, this comprises the start and duration of the execution, whether the test was deployable, the overall test result, and last but not least the number of all, successful and failed test cases. For performance

[10] The license is given according to the enumeration at http://spdx.org/spdx-license-list/license-list-overview.

benchmarking, the result comprises both performance (e.g., process duration, throughput, and number of process instances) and resource utilization (e.g., CPU, RAM, IO, and size of stored data) metrics providing the minima, maxima, average, standard deviation and the relative standard deviation.

The example on the right hand side of Fig. 2 describes a single test which evaluated the feature *SequenceFlow* on the engine *camunda BPM 7.3.0* with the benchmarking tool *betsy*. The feature is part of the *conformance* benchmark for *BPMN*, and within the group *basics*. The test configuration of the test comprises a single test case which passes two integers into the workflow and asserts that the trace of the executed workflow must contain *task1*. The test results describe that the test was deployable and successful.

All functional requirements from Sect. 3 can be fulfilled with the data from the common data model. We can compare the results (a) along the feature tree using either the feature level or any other aggregation level, (b) along the engine dimension using either the latest engine, multiple versions of the same engine or any other combination of engine instances, and (c) drill down to the test results and the actual engine logs.

To add new benchmarks along with their results to the common data model, one can simply instantiate the existing structure and reuse the currently defined result formats and options to define test configurations. If these do not suffice, one can create their own result format for a new capability, and provide custom test assertions, steps, and partners through extension. One limitation of this common data model is the fact that the feature tree uses three levels of hierarchy to group the testable features, and therefore one must adhere to that structure. In case of BenchFlow, which does not require any grouping of features, we introduced empty placeholder groups to fit the data model nonetheless.

6 Implementation

This section outlines the implementation of both, the transformer and the dashboard. The transformer is implemented in Java 8, whereas the dashboard relies on HTML5 and JavaScript alone. Both interact with each other through the exchange of the common data model which is implemented in JSON[11].

The **transformer** converts data from betsy and BenchFlow into the common data model. It does this by adding the data to an existing instance of the common data model, i.e., acting like an extract-transform-load (ETL) process known from the field of Business Intelligence with their Data Warehouses. However, in contrast to a Data Warehouse, it overwrites the results if a newer benchmark result is mapped to an existing one. During the mapping, the transformer ensures the uniqueness and resolvability of atomic and composite IDs, and manages the links to the external files. The transformer itself is usable through a command line interface, and open source as well as publicly available[12].

[11] See http://www.json.org/.
[12] See https://github.com/peace-project/transformer.

The **dashboard** is open source[13] and publicly available[14] as well. It contains seven pages: The *start page* provides links to the pages containing the actual benchmark results. Three *capability pages* have been created, each of them presenting benchmark results of different capabilities: for *conformance*, for *expressiveness* and for *performance*. The *engine compare* page allows users to compare two different engine instances by their features as well as by their benchmark results. The *engine overview* page allows the user to quickly get an overview of the engine instances and how each engine instance supports the different capabilities. Information about the project is included in the *about* page.

Both the transformer and the dashboard have been built for **extension**. The three steps to add a new capability with benchmarking results are as follows. First, the transformer must be extended with the ability to convert the data model of the new benchmark capability to the common data model as described in Sect. 5. Second, as each capability has its own result format, we need to provide a function which can interpret and load the benchmark results from the common data model. Third and last, a HTML template must be created which displays the benchmark results of the new capability. We can reuse the logic for filtering the data along the feature tree, but we must specify which filters and drill-down functions make sense.

7 Conclusion and Future Work

In this paper we have presented (a) a common data model for both workflow benchmarking frameworks betsy and BenchFlow, (b) a transformer which can map data from betsy and BenchFlow to the common data model, and (c) an interactive dashboard which uses the common data model to visualize the data so that end users, developers or researchers can compare workflow engines. The implementation of the dashboard was driven by requirements in form of user stories which are shown to be fulfilled with the implementation.

Future work is subdivided into technical and organizational improvements. Regarding technical ones, we aim to add charts and graphs to the dashboard so that the existing data is presented even clearer. Further, we aim to incorporate additional capabilities and benchmarks, so that we can present interesting metrics, e.g., the required effort to implement new features, how many bugs are filed over a period, and time until a bug is fixed. Last, we try to generalize the common data model even further to cover benchmarking results for other software systems as well. Regarding organizational improvements, our plan is to build up a community for workflow engine benchmarking comprising end users, developers, and engine vendors alike. This community can then drive the improvement of the dashboard, betsy and BenchFlow through their feedback.

[13] See https://github.com/peace-project/dashboard.
[14] See peace-project.github.io.

Acknowledgment. We would like to express our gratitude to Jörg Lenhard and Matthias Geiger for fruitful discussions and feedback regarding the dashboard, and both, Marigianna Skouradaki and Vincenco Ferme, for helping in bringing the BenchFlow data into this dashboard.

References

1. 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. IFIP AICT, vol. 451, pp. 81–90. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17837-0_8
2. Elias, M., Bezerianos, A.: Exploration views: understanding dashboard creation and customization for visualization novices. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011. LNCS, vol. 6949, pp. 274–291. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23768-3_23
3. Ferme, V., Ivanchikj, A., Pautasso, C.: A framework for benchmarking BPMN 2.0 workflow management systems. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 251–259. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_18
4. Few, S.: Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly, Massachusetts (2006)
5. Geiger, M., Harrer, S., Lenhard, J.: Process engine benchmarking with betsy – current status and future directions. In: ZEUS, pp. 37–44, January 2016
6. Geiger, M., Harrer, S., Lenhard, J., Casar, M., Vorndran, A., Wirtz, G.: BPMN conformance in open source engines. In: SOSE, March 2015
7. Geiger, M., Harrer, S., Lenhard, J., Wirtz, G.: On the evolution of BPMN 2.0 support and implementation. In: SOSE, pp. 120–128, March 2016
8. Harrer, S., Lenhard, J., Wirtz, G.: BPEL conformance in open source engines. In: SOCA, pp. 237–244, December 2012
9. 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
10. Harrer, S., Nizamic, F., Wirtz, G., Lazovik, A.: Towards a robustness evaluation framework for BPEL engines. In: SOCA, pp. 199–206, November 2014
11. Harrer, S., Preißinger, C., Wirtz, G.: BPEL conformance in open source engines: the case of static analysis. In: SOCA, pp. 33–40, November 2014
12. ISO/IEC. ISO/IEC 25010:2011; Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011)
13. ISO/IEC. ISO/IEC 19510:2013 – Information technology - Object Management Group Business Process Model and Notation, November 2013. v2.0.2
14. OASIS. Web Services Business Process Execution Language, April 2007. v2.0
15. Oppenheimer, D., Brown, A.B., Traupman, J., Broadwell, P., Patterson, D.A.: Practical issues in dependability benchmarking. In: EASY, p. 7 (2002)
16. Pauwels, K., Ambler, T., Clark, B.H., LaPointe, P., Reibstein, D., Skiera, B., Wierenga, B., Wiesel, T.: Dashboards as a service: why, what, how, and what research is needed? J. Serv. Res. **12**, 175–189 (2009)

17. Röck, C., Harrer, S., Wirtz, G.: Performance benchmarking of BPEL engines: a comparison framework, status quo evaluation and challenges. In: SEKE, pp. 31–34, July 2014
18. 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
19. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distribu. Parallel Databases **14**(1), 5–51 (2003)
20. WfMC. Terminology & Glossary. WfMC, February 1999. v3.0
21. Yigitbasioglu, O.M., Velcu, O.: A review of dashboards in performance management: implications for design and research. IJAIS **13**(1), 41–59 (2012)

A Distributed Cross-Layer Monitoring System Based on QoS Metrics Models

Damianos Metallidis(✉), Kyriakos Kritikos, Chrysostomos Zeginis, and Dimitris Plexousakis

ICS-FORTH, Heraklion, Greece
{metal,kritikos,zegchris,dp}@ics.forth.gr

Abstract. Monitoring of business process workflows based on metric quality models is associated to a gap between the definitions of workflow, service and infrastructure layer quality metrics. Most monitoring frameworks rely only on a layer-specific quality model, covering, e.g., the service layer, without considering the cross-layer dependencies it might have with quality models in the rest of the layers. The novelty of this paper closes the gap between the different functional layers by defining a cross-layer dependency model indicating relationships of quality aspects from three different semantic quality models. Each of these three quality models define metrics, metric aggregations and computations for each of the separate layers. These quality models are being addressed by a continuously, yet evolving distributed monitoring system.

Keywords: Quality models · Cross-layer dependencies
Quality metrics · Semantics · Computation · Monitoring · Aggregation

1 Introduction

In order to implement cross-organisational workflows and to realise collaborations between small and medium enterprises (SMEs) the use of Web service technology and Service-oriented Architecture (SOA) has become a necessity. Whilst, SMEs are continuously moving towards service-oriented infrastructures where applications are being modeled, the need of hosting them has raised an important issue for the quality of the underlying Cloud infrastructures. Virtualization, provided by cloud infrastructures, delegates the use of any kind of resources, such as computing environments or storage systems, to the data center's internal networks. All of the above issues raised the need for monitoring of the quality of the acquired resources and of the services offered to final users as also the workfload-based procedures used by SMEs in order to use services.

We address those aspects by specifying Quality Models (QMs) that a monitoring system could adopt so as to gather monitoring data taken from three different layers: (a) Workflow, (b) Service and (c) Infrastructure layer. State-of-the-art research is based mostly on individual layers without considering the cross-layer dependencies [17] that Quality Models (QM) might have. This leads

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 189–200, 2018.
https://doi.org/10.1007/978-3-319-72125-5_15

us to propose layer-specific QMs along with a cross-layer QM catering for dependencies among layers. Intention of QMs is the specification of quality terms (e.g., quality attributes) as well as of the respective relationships between these terms. In order to specify the legitimate structure of a QM, the respective conceptualisations and all possible types of quality term/concept relationships, a Quality Meta-Model (QMM) [14] should be in place. In this paper we use OWL-Q [14] as a semantic SQMM able to define semantic QMs. While different QMMs have been proposed using different representation formalisms, ontologies seem to be the best formalism as they provide a formal, syntactic and semantic description model of concepts, properties and dependencies between concepts. Moreover, they are extensible, human-understandable and machine-interpretable and enable reasoning via using Semantic Web technologies.

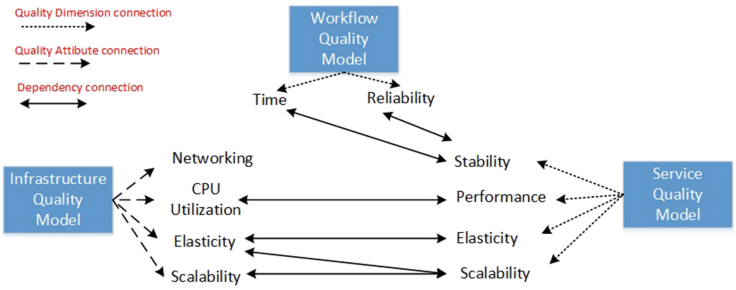


Fig. 1. Quality Models terms and cross-layer dependencies

Catering for the workflow-based usage of a service as well as the infrastructure that supports software components realizing part of the workflow functionality, we take into account three main QMs: (a) the workflow QM (WM) stressing quality terms related to tasks and workflows, (b) the service QM (SM) indicating quality terms for assessing the quality of services, and (c) the infrastructure QM (IM) encompassing quality terms suitable for the assessment of the quality of the underlying cloud infrastructure. Additionally, we have defined cross-layer relationships between quality metrics defined in the three QMs, in a sense that a metric defined in layer X can be used for the calculation of a metric defined in layer Y. Figure 1 depicts the quality models and terms that have been defined along with the cross-layer connection dependencies between them, indicating relations between quality metrics.

We are heavily focused on WM quality metrics, in a manner such that a possible user of any complicated workflow (we address this by an example of a multi-structure workflow, see Sect. 3.1) is able to evaluate results of procedures that include benchmarking and conformance tests based on the aforementioned aspects. Following this direction we will be able to deliver Workflow *Monitor* as a Service (W*M*aaS) in any Workflow as a Service (WfaaS) procedure. The definitions of the quality terms that are being proposed will help SMEs to give

performance and reliability grades on their business workflows, standardizing the way that the workflows are going to be evaluated.

The rest of the paper is structured as follows. In Sect. 2 we review the related work, while in Sect. 3 we represent the four QMs. In Sect. 4 we demonstrate the architecture of our monitoring system, whilst Sect. 5 provides conclusions and future work directions.

2 Related Work

As already mentioned, there are many layer-specific approaches for QMs. For instance, approaches in [12] and [7] are based on stochastic models and probabilistic theory having as a major concern the scalability of the Cloud resources based on quality metric results. Several approaches have been proposed capturing infrastructure QMs with focus on Cloud resources. Authors in [4] define QMs which support the evaluation of public Cloud services; the validation of these QMs is performed according to empirical case studies without taking into account the relations that WM, IM and SM can have between them. Precise definitions of scalability and elasticity are given in [5,11], spanning mostly the service and infrastructure layers; our work on the other hand defines scalability and elasticity metric dependencies between those two layers and does not cope with them in isolation.

In [6] a layer-specific extensive WM is proposed for the specification of workflow QoS, as well as methods to analyze and monitor QoS. In [9], the authors introduce the hypothesis that reliability of workflows can be notably improved by advocating scientists to preserve a minimal set of information that is essential to assist the interpretations of these workflows and hence improve their reproducibility and reusability, but with no user constraints taken into account.

A very interesting platform-independent solution has been proposed in [8] in order to support reconfiguration in service-oriented distributed soft real-time systems, in favor of time-bounded operations. As the main focus is on real-time monitoring and reconfiguration, the fact that the services might be a functional piece of a workflow has not been taken into account. This paper mainly involves service-oriented applications, which could help in the possible re-organization of the services themselves in order to function properly.

Finally, state-of-the-art research provides some approaches towards cross-layer monitoring. Kazhamiakin et al. [13] define appropriate mechanisms and techniques to address monitoring in an integrated cross-layer framework. In [10] the authors present an integrated approach for monitoring and adapting multi-layered SBAs. The approach coprises four main steps: (1) monitoring and correlation, (2) analysis of adaptation needs, (3) identification of multi-layer adaptation strategies and (4) adaptation enactment. Finally, in our previous work [17] we propose a monitoring framework for Multi-Cloud SBAs, that is able to perform cross-layer (Cloud and SOA) monitoring enabling concerted adaptation actions.

3 Definition of Quality Models

Quality dimensions cover an important aspect [14] of QMs involving dimension-specific attributes that can be measured by one or more dimension-specific metrics. Calculation formulas, quality metrics and particular types of metric relationships are defined in a formal way to formulate the structure of the respective layer. To achieve this, we have as initial guide quality dimensions/attributes, that assist in finding candidate metrics at a lower layer which could be connected/mapped to metrics at the higher layers. To produce suitable and complete QMs, which can be easily exploited by cross-layer monitoring systems, the quality terms included in them should satisfy the properties of measurability, validity and definition formalization [16]. Before we continue with the definition of the QMs to be used by a cross-layer monitoring system [17], we should also mention that these QMs rely on characteristics, stated in [15] (using OWL-Q), that each of the quality metrics should have.

3.1 Workflow Quality Model (WM)

We have defined a QM for the workflow layer, which contains workflow and task metrics as advocated in [6]. It also explicates how task measurements can be propagated to the level of workflow to produce the respective measurements of workflow metrics through metric aggregations. The proposed QM comprises the quality dimensions of *time* and *reliability*.

Time quality dimension of time is a fundamental aspect of performance that describes the time needed in order to measure, execute, record, respond and traverse through operations.

Concerning the Workflow level we have defined metrics of *Workflow Process Time* (WPT), which is calculated by the addition of the *Workflow Execution Time* (WET) and *Workflow Delay Time* (WDT) over all tasks along with the *Workflow Transition Delay Time* (WTDT) from one task to another. In addition, we have defined *Overall Workflow Execution Time (OWET)* indicating the overall execution time spent when executing all tasks in the workflow being independent from the workflow structure.

Calculation of Workflow Execution Time. To assist in the calculation of WET we have defined a composite metric called *sub-WorkflowExecution* (sub-WE) which represents the execution time of a workflow's sub-structure/sub-element. Possible values of the $sub - WE$ metric depend on the type of the respective sub-structure. A sub-structure could be any sequential or parallel structure within a workflow (we neglect conditional ones as they are being possessed at run-time). In the case of a sequential structure, the respective sub-WE metric is computed from the addition of Task Execution Time *TExT*, defined as the amount of time spent to perform the task by any entity (e.g., a human-based or software component), and sub-WE values mapping to the contents of this structure (i.e., tasks and internal sub-structures), respectively. In case of a parallel structure, the sub-WE metric depends on the max execution time value over the path branches included in this structure.

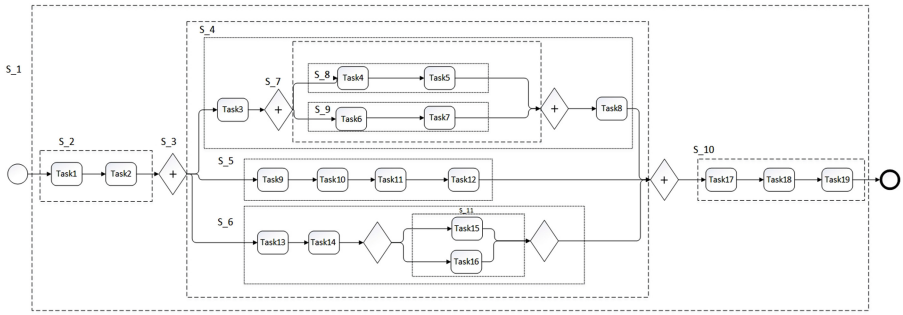


Fig. 2. BPMN Workflow indicating sub-WE elements

A new sub-WE is defined in three main situations: *(a)* For the single global/outer structure of the workflow which is equal to the workflow's WET (S_1 in Fig. 2), *(b)* When next step in the workflow is a parallel structure and *(c)* when next step in the workflow is a sequential structure. A sub-WE is going to be computed once it's component values/measurements are available i.e., the encompassing sub-WE and TExT metrics.

Figure 2 depicts a nice and slightly complex BPMN workflow model which can reflect the reality and is mainly used to explicate the computation procedure for workflow metrics. This model includes eight sequential structures, two parallel structures, and a conditional structure. S symbol represents any kind of structure in the BPMN diagram. Below we represent the formulation of our algorithm along with an example following that procedure.

Table 1. Formulas for the calculation of WET.

Metrics for the calculation of the WET	Calculation formulas
Workflow Execution Time (WET)	sub-WE_1
sub-WE_i	Sequential Element Case: $TExT_1$ + + $TExT_N$ + sub-WE_2 + + sub-WE_K Parallel Element Case: max(sub-WE_2,...,sub-WE_K, $TExT_1$,...,$TExT_N$)

In Table 1, i indicates a number between 1 and K, where K is the number of sub-structures which maps to sub-WE metrics and N is the number of tasks which maps to the TExT metrics.

As shown in Table 1, we rely on a procedure which leads to the production of a simple computation formula. This formula is recursively broken down into additional components, so as to compute the WET metric of a specific workflow. To compute each of the sub-WE metrics we have the convention that sub-WE_1

represents the execution time spent in structure S_1, sub-WE_2 represents the execution time spent in structure S_2 and so on. Spanning workflow tasks, $TExT_1$ represents the execution time spent in Task1, $TExT_2$ represents the execution time spent in Task2 and so on. Thus, following the above rules and Table 1 formulations, WET equals to sub-WE_1, sub-WE_1 equals to the addition of sub-WE_2, sub-WE_3 and sub-WE_{10}. Sub-WE_2 equals to the addition of $TExT_1$ and $TExT_2$. Sub-WE_3 equals to the max (sub-WE_4, sub-WE_5, sub-WE_6). Same calculations are done for the remaining numbers of the substructures in order to get the right values. The calculation procedure of $WTDT$ and WDT metrics is similar to WET.

Reliability quality dimension corresponds to the likelihood that a component (e.g., workflow, task, service) will not malfunction or fail. We have defined the following workflow metrics:

– *Workflow Fidelity (WF):* It measures the satisfaction of the workflow instances over the user quality requirements, within a specific time period. Thus, we can measure WF by applying the values of workflow metric measurements (mapping to the specified period of time) to a utility function which depends on users requirements.

Additionally we define the following task metrics:

– *Task Fidelity (TF):* It computes how well tasks instances meet user requirements (at the task level) within a specific time period by utilizing similar functions with respect to the case of WF.

Table 2. Calculation formulas of Workflow and Task metrics.

Workflow and task metrics	Calculation formulas
Workflow Fidelity(WF)	$f_{WF}(hist, reqs)$
$f_{WF}(hist, reqs)$	$\frac{(sat(meas_1, reqs)+...+sat(meas_A, reqs))}{A}$
$sat\,(meas_i, reqs)$	if $reqs.lowThreshold \leq meas_i.value$ $\wedge$ $meas_i.value \leq reqs.upperThreshold$ return 1; else return 0;
Task Fidelity(TF)	$f_{TF}(hist, reqs)$
$f_{TF}(hist, reqs)$	$\frac{(sat(meas_1, reqs)+...+sat(meas_B, reqs))}{B}$

In Table 2, we define the Workflow Fidelity formula. This formula maps to the function $f_{WF}(hist, reqs)$, which takes as input two parameters. The first parameter is a set of metrics along with their measured values, mapping to the interested time interval; while the second parameter represents the user requirements. A measurement is composed of a metric, a value and a timestamp, whilst each user requirement represents a threshold being applied over a specific metric. This function has a body, which computes the mean satisfaction level of user requirements over the

considered measurements represented by a value in the range [0.0, 1.0]. The satisfaction level for each measurement is computed from the *sat* function which initially selects those user requirements which map to the respective metric of the measurement and then performs the comparison of the measurement value against the user requirement low and upper bound/thresholds. If the metric measurement is within these thresholds, the output is 1; otherwise, it is equal to 0. In the case of tasks, we independently calculate their reliability in the context of a particular workflow, by considering a specific time interval and similar calculation functions to the workflow fidelity.

3.2 Service Quality Model (SM)

We have defined a SM based on metrics that assesses the quality of the respective service. Quality of a service maps to the quality that a requester perceives when using this service based on Service Level Objectives (SLOs) but also to the internal quality of this service which maps to the capturing of the service provider view. We can rely on *performance* metrics, indicating the performance of a web service; on *stability* metrics that are related to the service *reliability* and *availability*.

The *performance* dimension refers to the velocity of a service responding to any service request. It can be described by metrics that refer to the quality attribute of *Response Time*, such as *(a) Request Completion Time*, which depicts the time point that all the data mapping to a response arrive at the user, *(b) Execution Time* indicating the time taken for the service to execute a single request and *(c) Delay Time* defining the delay time of the software component in order to start processing the request.

Stability quality dimension indicates the ability to provide reliable, continuous, consistent and recoverable services despite undesired situations like increased load, congestion, system failure and natural disasters. This quality dimension has the following quality attributes:

- *Availability:* We define availability as the ratio of time in which a service is expected to function properly. We have to stress that we do not consider networking issues as these are related to service accessibility. Service clients or third-parties assess the availability of services based on the uptime status (as conceived by these entities which could also be related to network issues that cannot be easily detected). Thus, we have defined the metrics of *Down Time* which is the total time that the service is not available and the metric of *Availability* indicating the external availability of the service as being viewed by external services. *Availability* equals to $1 - \frac{DownTime}{TotalTime}$, where $TotalTime$ is the total observation time.
- *Reliability:* The reliability quality attribute measures how reliable is the service reffering to the fact of having the least possible number of failures and the largest possible amount of time between them, according to user constraints. Metrics of this attribute are *Mean Time To Failure* (MTTF), *Mean Time Between Failures* (MTBF) and *Fidelity* as also being defined in WM, but being adapted for the notion of SM.

By relying on [11], service *scalability* is defined as the ability of a service to scale and satisfy the agreed SLOs, when additional workload is received. The metrics defined for realizing service scalability are the *Scaling Utilization* metric which assesses the percentage of time a service exploits more or less resources than needed and *Scaling Precision (%)* which computes the percentage of time where the scaling of a service process was successful and according to the agreed SLOs, by dividing the number of successful scaling actions by the total number of scaling actions.

For the QM of Service layer we have defined the *elasticity* quality dimension as the degree at which a service system can autonomously scale the amount of service instances based on workload fluctuations to still conform to the SLOs agreed. A service's elasticity can be computed by the metrics of *Mean Time Taken to React* ($MTTrct$) and *PerfScaleFactor*. Metric of $MTTrct$ is defined as the mean time of reaction from the moment the need of scaling is detected until the respective scaling is completed, which is derived from the metric of *Reaction Time* indicating the raw time that the reaction takes. In addition, $PerfScaleFactor$ is the scale factor of the performance between two invocations of the same service before and after the scaling has been performed; equals to $\frac{\sum_{i=1}^{N} perfscalefactor_i}{N}$, where $perfscalefactor_i$ is the scale factor of i metric and N is the number of metrics being considered.

3.3 Infrastructure Quality Model (IM)

At the infrastructure layer we use four quality attributes defined in [4], as depicted in Fig. 1 and define the corresponding metrics. Mainly *Network as a Service* (NaaS), *Platform as a Service* (PaaS) and *Infrastructure as a Service* (IaaS) is the reference point of most of the quality metrics defined.

Networking quality attribute characterizes the quality of a data center's (DC) network. In order to assess internally the network performance, we cover the DC operator and the SaaS provider. Both care about the DC network performance (e.g., an application execution may involve invoking different components situated in different DC VMs). We define the respective metrics of *(a) Mean Packet Loss Frequency (lost packets/min)* indicating the mean rate of lost packets that failed to arrive at their destination, *(b) Max Connection Error Rate* which defines the maximum rate at which connection errors occurs and *(c) Packet Transfer Time* indicating the mean packet transfer time from/to its source/destination accordingly.

Quality attribute of *CPU Utilization* depicts the level at which processors are leveraged within a cloud infrastructure in favour of a client. Thus, we have define the metrics of *(a) Arrival Rate (transactions/ms)* indicating the workload that is arriving at the CPU in certain point of time *(b) Shared Physical CPUs* indicating the number of different VMs function on each of the CPUs *(c) Network shared Physical CPUs* which is the number of different VMs that the physical CPU is being used of but between different network clusters *(d) Virtual CPUs* indicating the number of virtual CPUs that are being served by each of the physical CPUs

and scheduled by the according hypervisor *(e) CPU Average Load (%)* which indicates the average CPU load for a processor, *(f) CPU Overall Maximum Load (%)* which indicates the overall maximum CPU Load being monitored which is further calculated by the metric of *(f_1) CPU Maximum Load(%)* indicating the peak load of the CPU within a specific period of reference. Similar metrics are being defined in case of CPU Minimum Load.

PaaS/IaaS Scalability quality attribute is defined [11] as the ability of the underlying infrastructure to sustain increasing workloads by making use of additional resources, that are directly requested, including all the hardware and virtualization layers. Based on [5], we define *Scalability Range (ScR)* as the ability of handling maximum workload that can still be handled by the underlying infrastructure to still satisfy the corresponding SLOs.

To support the notion of elasticity we are also based on [11], where it is defined as the degree to which a Cloud infrastructure can adapt on workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each time point the available resources match the user requirements, as much as possible. In addition, we rely on precision [11] to define elasticity metrics which is the absolute deviation of the according amount of allocated resources from the actual resource demand. Based on the aforementioned aspects of elasticity, we use the metrics of *(a) Precision of scaling out/scaling in* (P_O,P_I) and *(b) Mean Time To Quality Repair (MTTQR)* from [5,11], respectively.

3.4 Cross-Layer Dependency Quality Model

To formalize relationships between quality metrics across the three layer-specific QMs, we have considered an initial set of cross-layer dependencies in the form of a dependency quality model. These dependencies indicate (a) that the computation of a metric on layer X can be used on layer $X + 1$ to complete the computation of a relevant metric and (b) the waxing dependencies that might exists between different layer metrics. Relevance could map to either both metrics belonging to the same quality dimension, or being described by similar quality attributes. Via the cross-layer dependencies (Fig. 1), the metric aggregation formulas and the fact that raw quality metrics with no dependencies can be calculated by sensors placed by the distributed monitoring system on one of the respective layers, the measurability of all metrics defined is guaranteed.

We have separated the cross-layer dependencies between the metrics by considering groups of adjacent layer-specific QMs:

- For the *SM* and *WM* group the following dependencies have been captured:
  - *Task execution time* of a service task in a workflow can be computed from the execution time of the service used to realise this task's functionality.
  - A service task's fidelity equals the fidelity of the service realising its functionality, when correspondence between *task* and *service* component is valid.
  - Metric of *Task Delay Time* defined in WM can be used in order to derive the value of the *Delay Time* of a service component defined in SM.

– Next group of dependencies derived are from the SM and IM QMs:
  - *Mean Time to Quality Repair* defined in IM has an *equality* reference to SM for the *MTTrct* defined in *elasticity* dimension. Thus, if we are referring to the same re-actions due to workload changes, then we can derive MTTrct of SM by mapping the workload changes to actions that had an impact on the scalability of the service.
  - There is an increasing trend that relates *Scaling Utilization* defined in SM and *Scaling Range* defined in IM. When utilization of scaling is high it gives us the sign that the underlying infrastructure is capable of handling scaling actions in high rates, meaning that the scaling range has also an increasing value making them waxing dependent values.
  - Regarding *CPU utilization* defined in IM we can infer that is a waxing dependent value with *response time* defined in SM. CPU utilization is increased if the overheads associated with context switching are being minimized and happen infrequently, thus large values of *CPU Average Load* for the according process context. This will have as a result the increase of the execution time for the according services.

4 Architecture of the Cross-Layer Monitoring System

As Fig. 3 shows, Virtual Machines (VMs) in *Public Cloud 1* include monitoring sensors, metric calculation aggregators and database instances. There is more than one user VMs that compromise the IaaS and SaaS aggregators with the according sensors being deployed. Information being retrieved from the aforementioned user VMs is passed on the *WF Engine VM* through a publish/subscribe mechanism. The rationale of having only one VM that the *WF Engine* is established on, is that a certain Workflow Engine is responsible for the business processes for each of the public clouds. Thus, by passing information to the *Workflow VM* we compute and store values of metrics related to VMs of the according Cloud. The role of the *WF Engine VM* is not only to have a *WF Engine aggregator*, but also a Cloud-dependent aggregator that is responsible for (a) aggregation of metric values based on the cross-layer dependency QM and (b) the measurement propagation to the user VMs in order to fulfill missing cross-layer dependencies that might exist on the according user VMs.

The next step is to pass the metric dependency data, through the publish/subscribe mechanism, on our private infrastructure. Then, we store and relate metric measurements being published from different public clouds based on the QMs that we have defined. Thus, by inter-correlating metric measurements and forcing the propagation of them across public clouds we are fully aware of the functional aspect of the according services.

Monitoring tools which are being used in order to implement and extend [17] the functionality of the system are the monitoring tools of Prometheus [3] for SM, Nagios monitoring tool [2] for IM and the monitoring capabilities of Activiti [1] in case of WM. The provisioning process of the VMs in each of the public clouds sectors are out of the scope of this paper, nevertheless having a major impact on the performance of PaaS in IM.

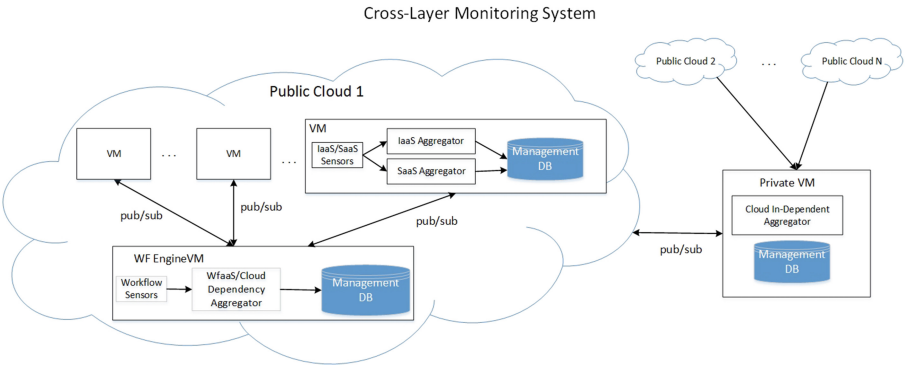


Fig. 3. Physical architecture of the Cross-Layer Monitoring System

5 Conclusions and Future Work

In this paper we have demonstrated three metric quality models along with a fourth one indicating the cross-layer dependencies and relations between quality metrics of the three QMs of WM, SM and IM. QMs that have been proposed are being covered by defining quality terms describing each of the layer's metric composability. To address that, we have created computation formulas that can be used to assign values on cross-layer depended-metrics, that could not been calculated unless dependencies are not defined. As for future work, we are in the process of extending the distributed monitoring system framework [17] in order to realize the support for the QMs proposed. Furthermore, we are going to enrich our QMs and our dependency model, so as to consider additional aspects (like security) mapping to the consideration of new domain-independent metrics. We are going to consider metrics in order to: *(a)* better validate the approach according to a specific use case; *(b)* highlight that the current structuring of the QMs is appropriate/suitable.

Acknowledgments. This work is supported by (*a*) CloudSocket project (http://www.cloudsocket.eu) that has been funded within the European Commissions H2020 Program under contract number 644690 and (*b*) PaaSage (http://www.paasage.eu) (FP7-317715) EU project.

References

1. Activiti workflow engine. http://activiti.org/
2. Nagios monitoring tool. https://www.nagios.org/
3. Promitheus monitoring tool. https://prometheus.io/
4. Bardsiri, A.K., Hashemi, S.M.: Qos metrics for cloud computing services evaluation. Int. J. Intell. Syst. Appl., 27–33 (2014)
5. Becker, M., Lehrig, S., Becker, S.: Systematically deriving quality metrics for cloud computing systems. In: John, L.K., Smith, C.U., Sachs, K., Llad, C.M. (eds.) ICPE, pp. 169–174. ACM (2015)

6. Cardoso, J., Miller, J., Sheth, A., Arnold, J.: Modeling quality of service for workflows and web service processes, October 2002
7. Cardoso, J., Sheth, A., Miller, J.: Workflow Quality of Service. Technical report, LSDIS Lab, Computer Science, University of Georgia, Athens GA, USA, LSDIS Lab, March 2002
8. Garca-Valls, M., Lopez, I.R., Fernndez-Villar, L.: iland: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. IEEE Trans. Ind. Inf. **9**(1), 228–236 (2013)
9. Gmez-Prez, J.M., Garca-Cuesta, E., Zhao, J., Garrido, A., Ruiz, J.E.: How reliable is your workflow: monitoring decay in scholarly publications. In: Castro, A.G., Lange, C., Lord, P.W., Stevens, R. (eds.) SePublica. CEUR Workshop Proceedings, vol. 994, pp. 75–86. CEUR-WS.org (2013)
10. Guinea, S., Kecskemeti, G., Marconi, A., Wetzstein, B.: Multi-layered monitoring and adaptation. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 359–373. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_24
11. Herbst, N.R., Kounev, S., Reussner, R.H.: Elasticity in cloud computing: what it is, and what it is not. In: Kephart, J.O., Pu, C., Zhu, X. (eds.) ICAC, pp. 23–27. USENIX Association (2013)
12. Joshi, K.P., Joshi, A., Yesha, Y.: Managing the quality of virtualized services. In: 2011 Annual SRII Global Conference, pp. 300–307, March 2011
13. Kazhamiakin, R., Pistore, M., Zengin, A.: Cross-layer adaptation and monitoring of service-based applications. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave -2009. LNCS, vol. 6275, pp. 325–334. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16132-2_31
14. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benbernou, S., Brandic, I., Kertsz, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. **46**(1), 1 (2013)
15. Kritikos, K., Plexousakis, D.: Requirements for qos-based web service description and discovery. IEEE Trans. Serv. Comput. **2**(4), 320–337 (2009)
16. Schneidewind, N.F.: Methodology for validating software metrics. IEEE Trans. Softw. Eng. **18**(5), 410–422 (1992)
17. Zeginis, C., Kritikos, K., Garefalakis, P., Konsolaki, K., Magoutis, K., Plexousakis, D.: Towards cross-layer monitoring of multi-cloud service-based applications. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 188–195. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40651-5_16

Rethinking Services (ResearCH)

Preface of ReSerCh 2016

1 The Need for Services

Since its early inception, Service Oriented Architecture (SOA) [3] has provided a vivid abstract representation for the basic mechanisms and parts by means of which complex software solutions can be shaped with a compositional and highly modular approach. The main concepts on which it has been designed [2] made this architecture suitable for many domains spanning from the development of software-intensive systems to the integration of heterogeneous information systems.

Although huge effort has been spent especially in the 2000s, services research seems to be facing a stall in recent years, focusing and mainly rotating around a distinct set of highly-researched topics (e.g., service composition) and neglecting extremely challenging aspects (e.g., service governance). This is actually in contrast with the need for services. For instance cloud computing [1] is based on concept of providing services at different levels (i.e., infrastructure, platform, software). Microservices [4] are gaining more and more attention as the way to build reliable, scalable, and flexible software solutions.

Goal of the workshop is to foster the discussion around a new service research agenda where both academics and industries are the key players. Based on the research done in the past, the analysis of successful stories and failures, the study of the impact obtained by the proposed solutions, new topics and challenges need to be defined.

2 Redefine the Service Research Agenda

With this premise, our gathering sought to clarify the dimensions across which newly-found research ideas may rotate and shape the future of services research. A number of high-impact topics were introduced and discussed during the workshop, ranging from fair- to trustless- to continuous-computing, to user-friendly design, and the definition of ecosystem of services. In addition, the need for investigating the computer science perspective and the business perspective is also arose. Service-Dominant Logic [5] and SOA need to talk to each other for the purpose of designing, implementing, and continuously deploying usable, efficient, and effective software services.

The workshop concludes that cross-disciplinary collaboration on these topics is fundamental as service based systems will have more and more importance in various domains: from emergency management to data analytic and logistic. Although the basic principles of SOA remain valid, academics and industries need to develop methods and solutions for enabling the development, deployment and governance of service-based systems with a unprecedented level of flexibility, scalability and speed of continuity. For instance, Big data calls for data-intensive applications able to deal with

a lot of information, while fog computing calls for applications able to deal with numerous, nomadic, and heterogenous devices.

Among the several contributions submitted and selected through a peer-review process, the authors of three of them have been invited to submit an extended version. The three papers that follow enter into the details of some of the discussions we had during the workshop highlighting how service research still deserve a lot attention in the upcoming years.

References

1. Liu, F., et al.: NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292). CreateSpace Independent Publishing Platform, USA (2012)
2. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference model for service oriented architecture 1.0 (2006)
3. Papazoglou, M.P., Georgakopoulos, D.: Introduction. Commun. ACM **46**(10), 24–28 (2003)
4. Richardson, C., Smith, F.E.: Microservices. From Design to Deployment. NGINX e-book (2016). https://www.nginx.com/resources/library/designing-deploying-microservices/
5. Vargo, S.L., Lusch, R.F.: Service-dominant logic: continuing the evolution. J. Acad. Mark. Sci. **36**(1), 1–10 (2008). http://dx.doi.org/10.1007/s11747-007-0069-6

Pierluigi Plebani
Damian A. Tamburri

Organization

Workshop Organizers

Pierluigi Plebani	Politecnico di Milano
Damian A. Tamburri	Politecnico di Milano

Technical Program Committee

Luciano Baresi	Politecnico di Milano, Italy
Antonio Brogi	Università di Pisa, Italy
Elisabetta di Nitto	Politecnico di Milano, Italy
Schahram Dustdar	TU Wien, Austria
Valerie Issarny	Inria Paris, France
Patricia Lago	Vrije Universiteit Amsterdam, The Netherlands
Winfried Lamersdorf	University of Hamburg, Germany
Cesare Pautasso	University of Lugano, Switzerland
Barbara Pernici	Politecnico di Milano, Italy
George Spanoudakis	City University of London, UK
Farouk Toumani	Université Blaise Pascal, France
Genoveva Vargas-Solar	CNRS, France

Continuous, Trustless, and Fair: Changing Priorities in Services Computing

Stefan Tai(✉)

TU Berlin, Berlin, Germany
tai@tu-berlin.de

Abstract. Services computing research and practice traditionally has focused on the objectives of business alignment, software systems interoperability and on leveraging the Web as a compute platform. Corresponding technology solution stacks and architectural styles have been promoted. Today, and probably for the next decade to come, different objectives are replacing these original ones and, correspondingly, different solution stacks and architectural styles are emerging. Most notably, challenges such as frequent delivery of service systems, decentralization and business disintermediation, and "socially aligned" service systems lead us to continuous computing, trustless computing, and fair computing – three major trends that we expect to become the driving force behind next-generation service systems. In this paper, we discuss these trends and identify major research directions to deliver on these changing priorities.

Keywords: Continuous computing · Trustless computing
Fair computing

1 Introduction

After 15+ years of research and practice the services computing community is undergoing a fundamental change: newer technology is replacing older technology, research efforts of the past without any significant impact to-date are discontinued, and new research challenges are appearing. In this invited paper, we argue that the primary objectives of services computing are changing – from business alignment, software interoperability and web computing initially to continuous computing, trustless computing, and fair computing today and tomorrow – and that the community consequently must part from older themes and instead focus on addressing current and future priorities.

2 Services Computing – A Brief Historical Sketch

In the early 2000s, a time where XML was popular and interoperability of heterogeneous software components and systems was a priority objective, WS-* was

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 205–210, 2018.
https://doi.org/10.1007/978-3-319-72125-5_16

born. SOAP and WSDL, along with the manifold WS-* specifications addressing all kinds of enterprise concerns, were promoted as industry standards pushed by large corporations. Correspondingly, the services computing research community explored service systems from a variety of angles related to the primary objectives of software interoperability and web computing, driven by the idea of establishing a rich computing model based on the services abstraction to well-align IT services and business services. In the mid- and late 2000s, themes including service discovery, (business process-driven) service composition, and service semantics were on the research agenda, with WS-* being the natural choice for proof-of-concept and implementation.

At around the same time, REST emerged as the more lightweight computing model and alternative to WS-*. REST is an architectural style using ubiquitous, foundational web technology like HTTP. REST thereby defines architectural constraints but, unlike WS-*, does not introduce new technology standards. Less motivated by interoperability concerns especially of large IT corporations, REST focuses on leveraging the Web and its manifold resources for purposes of services computing.

Today, especially from a research perspective, WS-* is mostly history and best remembered as a set of XML specifications, which encode proven principles of enterprise computing, but which also tend to (invite to) introduce computational overhead. While SOAP and WSDL are still in use in many enterprise systems, the majority of the WS-*-specifications did not have any significant practical impact and – due to their focus on standardizing interoperability concerns – need no longer be subject of current or future research.

REST, on the other hand, today is by far the more popular services computing model. A large body of best practices is available, making REST a commonly applied and principally well-understood computing model. Few if any critical REST-specific research challenges are left open.

A third, more recent trend in services computing are microservices. Driven by the need to ease change management, microservices must be seen in the context of DevOps-based organizations: they directly link the software service artifact to the development and operations team that builds and manages the artifact – an aspect that both WS-* and REST have ignored – and emphasize communication between different teams by means of APIs. Like REST, microservices are not about standards, but about architecture. Unlike REST, microservices promote an architectural decomposition into individual business functions, where each business function may be a full vertical cut across multiple system layers including the data and resource management layer. Microservices thereby loosely couple the business functions, but tightly couple business logic and data management.

Yet another difference lies in the deployment and runtime environments that the three services computing models propose: WS-* advocates a traditional enterprise middleware environment, REST relies on the web itself, and microservices lend themselves naturally to cloud systems, especially deployments using container technology. Microservices can be seen as a native cloud-based services

computing model, whereas WS-* and REST describe models that were originally developed and proposed prior to the cloud-era.

3 Some Lessons Learned

WS-*, REST, and microservices describe three different services computing models. WS-* is mostly history, REST is current engineering practice, and microservices, along with related concepts of lambda-services and serverless architectures, are gaining momentum. Before we discuss future research directions, we can conclude:

1. Services computing fundamentally is about architecture – "making non-trivial decisions that are documented and are based on a clear rationale" [4]. Such decision-making is influenced by the services computing model chosen and the corresponding objectives associated with the model. The architectural principles are what drives and distinguishes service systems.
2. Architecture does not need complex and rich technology standards; the most basic and simple standards suffice. The engineering, the proposition and the use of rich standards, correspondingly, is not critical to service architectures and need not be on the services research agenda.
3. Architectural constraints change as the service engineering culture and technology evolves. With WS-*, interoperability was a priority objective. REST put web principles to the front. Microservices emphasize ease of change management. With different priorities in mind, different architectural solutions have been and will continue to be born.
4. Future services computing models will natively reflect advancements in computing infrastructure.

4 Research Ahead: Changing Priorities

We observe three major trends in services and cloud computing, which each replace former thinking and former priorities with newer thinking and newer priorities: *continuous computing*, *trustless computing*, and *fair computing*. These three trends, individually and in combination, reflect changing needs: frequent, i.e. almost 'continuous' delivery of systems, disintermediation of businesses and decentralized applications, and "social alignment" beyond "business alignment".

4.1 Continuous Computing

Continuous computing emphasizes the need to continuously, i.e., frequently, deliver a system. Continuity requires new engineering processes for delivery and a high degree of automation with appropriate tooling, along with organizational models that support these processes. In practice, "reducing the time between committing a change to a system and to place the change into normal production, while ensuring high quality" [1] – multiple times a day – is a critical

requirement. Architectural abstractions in support of an effective change management consequently are a top priority. We can identify at least the following main research challenges:

- *Engineering microservices-based architectures.* Microservices, born out of DevOps-based organizations with continuous delivery pipelines, and related concepts of lambda services, describe different building blocks for service-oriented architectures than the traditional WS-* or REST services. Their tight integration with organizational and delivery aspects makes them a natural candidate to support continuous computing. An integrated approach to the design and use of microservices, their management in cloud-based deployment and runtime environments, and their role in the organizational context and continuous delivery processes is required.
- *Cloud service benchmarking.* Frequent changes and continuous delivery require evidence-based quality control and management. With cloud service benchmarking, we refer to recurring quality-oriented experimentation and analysis of services deployed in cloud environments, for the purpose of discovering quality insights otherwise unknown [2]. Cloud service benchmarking, in addition to functional testing and monitoring of production systems, is critical to understanding service systems and to both justify and guide system changes in continuous computing.

4.2 Trustless Computing

In the past years, much attention was paid to trust in computing and trust models for services and cloud systems. This was largely driven by fear or risk aversion when outsourcing computing and data to external service and cloud providers. Trustful computing then suggested architectures that require complex security protocols, intermediation and, typically, some central authority to manage and/or warrant 'trust'.

Trustless computing deliberately breaks with such thinking and promotes decentralized solutions for the correct execution of 'transactions'. Unlike transactions as known from database systems, in trustless computing, no transaction manager and no concurrency and coordination control exist, but symmetric shared responsibilities, including transaction validation, by any node participating in the network. Peer-to-peer systems employing decentralized consensus protocols remove centralized control and allow for new forms of business disintermediation. Note that the term 'trustless' does not imply a lack of trust, but similar to the terms 'stateless' or 'serverless' in services computing, a change in perspective in how trust (or state, or servers) is managed.

To this end, one major research challenge stands out: *Blockchain-based application architectures.* Blockchains are decentralized, immutable ledgers for verifying and recording 'transactions'. Originally proposed along with the bitcoin cryptocurrency [5], blockchains today are the prime candidate solution for trustless computing in any application domain where trade occurs, and whenever trust is to be ensured through peers, but not by some central authority. Blockchains

currently experience intensive debate and hype. We agree that there is a huge potential associated with blockchains to disruptively change entire application domains, but argue that a careful selection of application domains and much more experimental research is needed. Blockchain-based applications are inherently distributed systems, and a distributed systems perspective is fundamental to building applications using blockchains. Solutions to deal with the typical fallacies of distributed computing are needed.

4.3 Fair Computing

Third, we observe that services computing no longer is driven by business thinking alone, but increasingly also by aspects of social awareness and social responsibility. For example, complex challenges such as *privacy* go well beyond business concerns but must focus on the human individual or group as the main stakeholder. We refer to this trend as *fair computing*, deliberately calling out for a modern computer science notion of *fairness* that may draw from fairness as studied in other scientific communities, especially law and economics. Research in fair computing demands at first two strands:

- *Fair Information Practices.* Different fair information practice principles have been around for decades, including those published by the US Federal Trade Commission [3]. These may serve as a first step and as general guidelines for fair computing in today's and tomorrow's service systems – covering aspects of transparency, choice and consent, and information review, correction and protection. Nevertheless, we expect refinements to be necessary as digitization continues to transform every aspect of life with unprecedented speed and impact. Privacy is more a "social alignment" challenge, complementing the general "business alignment" objective that services computing traditionally has focused on.
- *Trade-off management.* Dealing with complex challenges such as privacy inherently induces dealing with conflicting objectives within such challenges. Typical trade-offs relate to 'anonymity versus accuracy' or, from a distributed systems perspective, '(desired) security (levels) versus (acceptable) performance (impact)'. In addition, 'fairness' itself is often regarded to be in a trade-off relationship with 'efficiency', typically, in the context of resource allocation problems. Balancing and overcoming trade-offs at different levels of abstraction is hardly possible in a generic way, but typically requires system/application-specific exploration that is evidence-based using quantifiable objectives and corresponding benchmarking methods.

5 Next Steps

The three trends of continuous computing, trustless computing, and fair computing described above share significant commonalities. First, core principles of peer-to-peer computing are prominent in all three trends. Second, all three trends

are potentially highly disruptive in nature, replacing older technology stacks and former architectural thinking with different technology stacks and newer thinking. Third, they all build on a notion of a 'distributed service', where each service is tightly associated with critical, non-technical responsibilities – organizational aspects in continuous computing, independent validation in trustless computing, and compliance to fair practice principles in fair computing.

We expect architectural styles that define trend-specific sets of constraints to continue to emerge, and so will innovations and technology in support of all three trends. We do not expect a need to devise complex standards and standardization activities for such architectures, neither protocols or infrastructure, as long as fundamental architectural constraints and governing principles are agreed upon.

The services computing research community must re-focus by putting traditional and 'solved' (or 'failed') research topics aside, and instead focus on current and future priorities that are at the core of next generation service systems. Continuity of service delivery, decentralization, and fairness should move into the center of our attention.

References

1. Bass, L.J., Weber, I.M., Zhu, L.: DevOps - A Software Architect's Perspective. Addison-Wesley, Redwood City (2015)
2. Bermbach, D., Wittern, J.E., Tai, S.: Cloud Service Benchmarking. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55483-9
3. Federal Trade Commission: Privacy online: Fair information practices in the electronic marketplace (2000). https://www.ftc.gov/reports/privacy-online-fair-information-practices-electronic-marketplace-federal-trade-commission
4. Hohpe, G.: 37 Things One Architect Knows about IT Transformation. Leanpub, CreateSpace Independent Publishing Platform (2016). https://www.amazon.de/Things-Architect-Knows-About-Transformation/dp/1537082981
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008). http://bitcoin.org/bitcoin.pdf

Data Integration and Quality Requirements in Emergency Services

Chiara Francalanci and Barbara Pernici(✉)

DEIB, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milan, Italy
{chiara.francalanci,barbara.pernici}@polimi.it

Abstract. The requirements and an initial architecture for supporting the analysis of social media contents in emergency management tools are discussed. The use of social media to improve emergency maps and to support early warning is discussed, and the emerging requirements are outlined. The proposed architecture is designed to support enhanced Emergency Copernicus services, as proposed in the European project E^{2}mC (Evolution of Emergency Copernicus services). In the project a Witness component is going to be developed, to support social media monitoring and analysis and federated crowdsourcing.

Keywords: Emergency services · Social media · Map improvement
Data quality

1 Introduction

Emergency response activities are based on many support services. In this paper we focus on emergency systems providing tools that leverage social media in emergency management processes.

A classification of social media use in crisis response has been proposed in [6], where messages are classified in the following categories:

1. Requests for assistance by victims.
2. Distribute official warnings.
3. Establish situational awareness.
4. Sharing media (e.g., pictures) to assist in damage estimation projects.
5. Possibility of direct engagement among users, citizens, organizations, responders (not only to share information).

In the early phases of an emergency, in most cases it is important to exploit all possible information sources is a very short lapse of time, therefore an automatic analysis of these messages can provide further support in the emergency management activities. An initial proposal of a system for extracting information from

This work is partially based on the E^{2}mC project proposal. The authors acknowledge the contribution of all partners, and in particular of e-GEOS SpA.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 211–218, 2018.
https://doi.org/10.1007/978-3-319-72125-5_17

tweets in emergency situations was put forward in the TORCIA project [4,5], funded by the Lombardy Region, to manage information originated by tweets and transforming it into useful information, such as, for instance, early warning signals for events and identifying consequences of events on the infrastructures, providing their geolocalization.

A further step in this direction is planned in the E^{2}mC H2020 European project, starting in November 2016, which aims to demonstrate the technical and operational feasibility of the integration of social media analysis and crowdsourcing within the full Copernicus [1] EMS (Emergency Management System), which provides annotated maps as a product, starting from satellite and aerial images [3]. The main goals of the project are focusing on exploiting social media and crowdsourcing to improve the Mapping and Early Warning service components of the system, as illustrated in Fig. 1.

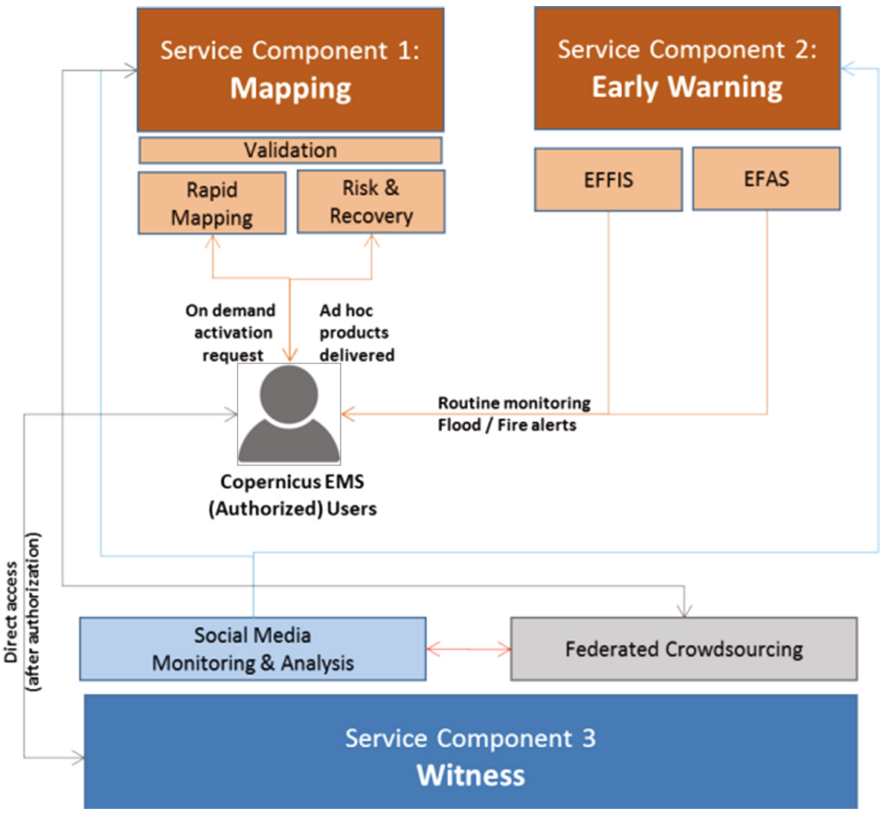


Fig. 1. E^{2}mC [2] extended Copernicus Emergency Management Service

The architecture illustrated in the figure shows the three main service components of E^{2}mC [2]:

- *Mapping*: to produce annotated maps in near real-time about areas of interest. An example of map is provided in Fig. 2, showing a grading map that annotates an aerial image with the severity of the damage to buildings and roads.
- *Early warning*: services more and more focusing on the so called impact-based forecast, where an estimate of the potential impact is provided together with the location and severity of the forecasted event.
- *Witness*: a new component to be developed in the project, to collect, integrate, and curate information from multiple social media sources and crowdsourcing.

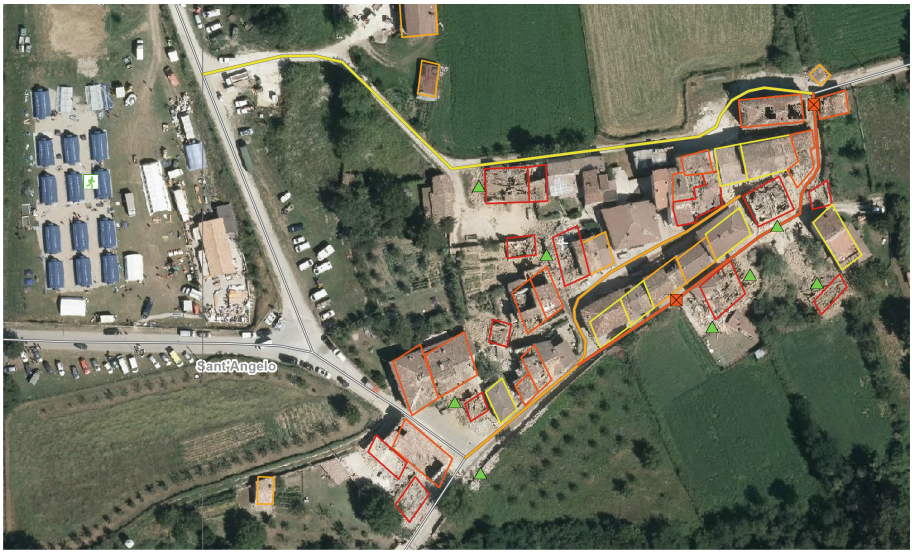


Fig. 2. Detail of an aerial grading map from emergency.copernicus.eu

The goal of this paper is to discuss the main components to be studied in E^{2}mC and the focus will be on illustrating research challenges of the Witness component.

The project is characterized by the need for integrating or at least relating several different sources of information (existing maps, satellite, social, crowd). The social media, in addition, provide a number of different types of information (text, sound, images, videos). As shown in a recent analysis of two emergency events described in [8], 50% of the tweets contain a link with a URLs. As a consequence, the analysis needs to go beyond the analysis of the text of the tweets, providing also tools to understand and possibly exploit also associated information. As a consequence, first there is the need to build an integrated knowledge

base, and then to exploit its contents as a basis for supporting the improvement of the quality of the maps derived from satellite images, in a semiautomatic way.

The Witness service is going to be developed as an independent service, to be connected to the other existing services mentioned above, but also to any other service operating in this domain, as an independent component. In Sect. 3, we illustrate the main components for information extraction and analysis and underlying design choices for a flexible architecture for E^2mC.

While the quantity of information posted on Twitter is large, the quality of tweets is often low. In a situation of emergency, the institutions and the individuals involved need precise information that can be practically helpful and make their work more efficient and effective. In E^2mC, information is useful if it can improve at least one the two fundamental services of Copernicus, that is early warning and rapid mapping. In both cases, the ultimate goal is to improve maps, by providing more timely and complete information. To this aim, the information collected from social media needs to be:

- recognized and interpreted, to guarantee that all required contextual information is available;
- validated, to guarantee dependability;
- integrated, to avoid redundancy and identify spamming.

For example, a picture has to be recognized and associated with a specific point of interest, has to be validated as a trustworthy representation of the emergency situation in the corresponding geographical position and, finally, it has to be grouped with other pictures providing the same type of evidence. These activities cannot be always and exclusively carried out by an automated tool. For example, a picture may represent a building, but the tool has no reference images of that building, or a picture may be posted without geotagging information and the tool fails to associate it with a position. In these cases, a crowd of individuals who are willing to cooperate to improve the quality of social media information can be extremely helpful. In E^2mC, crowdsourcing will be aimed to:

- mitigate the risks associated with information integration;
- reduce error rates, interpreting and classifying information;
- improve the overall accuracy and dependability of technology's outputs.

2 Research Challenges and Potential Developments

Several research challenges are posed by a service-based architecture in this specific domain. The main requirements are quantity of data (additional data), its quality (live maps combined with social multimedia contents) and its timeliness (first crisis maps available within few hours), while satellite data are not always available and costly services might need to be requested from the satellite platform: there is a need to complement satellite information to improve quality and the irregular provision of information from the satellites.

2.1 Data Requirements and Quality Requirements

A fundamental goal of the E^{2}mC project is to integrate multimedia information from multiple sources. This goal raises a number of technical challenges that are both related to specific technical issues (such as the identification of a tiny portion of useful information among large volumes of generic data) and to the need of considering multiple technical issues in combination to provide a usable result (such as the integration of text processing, image processing, geotagging and crowdsourcing functionalities to prepare a database of images that can aid Copernicus services). These issues are made even more challenging by the near real-time requirements and critical situations in which the system operates. The proposed approach is to adopt a keyword-based approach, which facilitates a multilingual annotation, and to support automatic and semi-automatic annotation by operators by micro-tasks for data quality checking supported by the crowdsourcing platform. The service-based approach will be adopted, providing operators with services to rapidly define new keywords and their associations on controlled vocabularies, and to automatically rank data wrt relevance and quality, e.g., from "full trusted" class (input for live notification and early warning process) to "to be checked" one (input for the crowdsourcing task). In particular, quality evaluation will be performed as a two-step process:

1. Authentication of source as reliable: for example, institutional sources representing operators that have emergency management responsibilities and official sources of news will be recognized and analyzed separately.
2. Triangulation of content as valid: this will be performed as a combined effort of software and crowdsourcing.

2.2 Ad Hoc Workflows with Source Evaluation

Service ownership and management have to be fully understood, to coordinate the different operators, for access control, and to allow the Copernicus EMS Rapid Mapping Team to build appropriate workflows to streamline the exploitation of data flows provided by the Copernicus Witness. New challenges are raised by the need of preserving privacy of subjects, of considering the quality of the data gathered from the different sources, and to control the output in terms of timeliness and quality fit for use in a given situation.

3 Towards a Service-Based Architecture for E^{2}mC

The core component of the E^{2}mC software architecture is the platform. The platform is composed by a service portal and shared social media data. The service portal provides a set of core REST data management services and a set of REST micro-services that have local data and can access shared data through the data management services. Both types of services are used by the E^{2}mC applications: the Web application, the mobile application, the federated crowdsourcing application and the crawlers (Fig. 3).

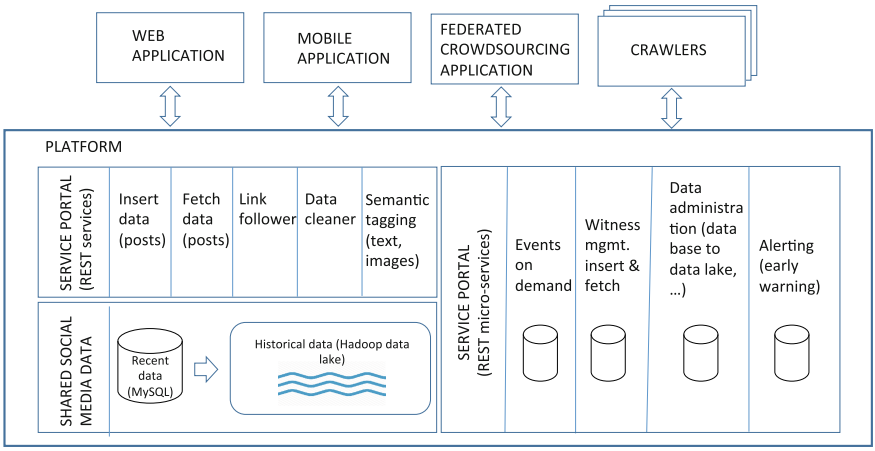


Fig. 3. E^{2}mC proposed software architecture

The main advantage of this architecture is the right balance between integration and flexibility, which are both needed to accommodate for changing requirements throughout the life span of the project. While core data and data management services are shared, that is, fully integrated, micro-services are independent of each other and can evolve according to different research and experimentation plans. For a same micro-service, different instances or versions can be simultaneously active and operate on different data for either software testing or field experimentation purposes. This trade-off between integration and flexibility is key to emergency management. A core set of data and related data management services is necessary to guarantee that Copernicus rapid mapping and early warning activities have access to integrated information that over time is enriched by different applications, such as crowdsourcing, in a consistent way. Given that social media information has a value, but is not always dependable, it is important that all applications work towards building a common set of consistent data. On the other hand, the micro-service approach gives applications the degree of flexibility that is important to support incremental development and experimentation by independent teams, towards separate goals, along different schedules. At runtime, micro-services can also be beneficial to integrate new functionalities as they emerge.

It should be noted that crawlers are considered an application, even if they are not a user application. This allows designing data management services that accommodate both the insertion of raw posts as well as partly or fully tagged posts.

The Lambda Architecture [7] shown in Fig. 4 will be deployed as the software infrastructure supporting data storage, data processing and service/micro-service provisioning. The main advantage of the Lambda Architecture is to support the deployment of Hadoop in a context requiring both big data storage &

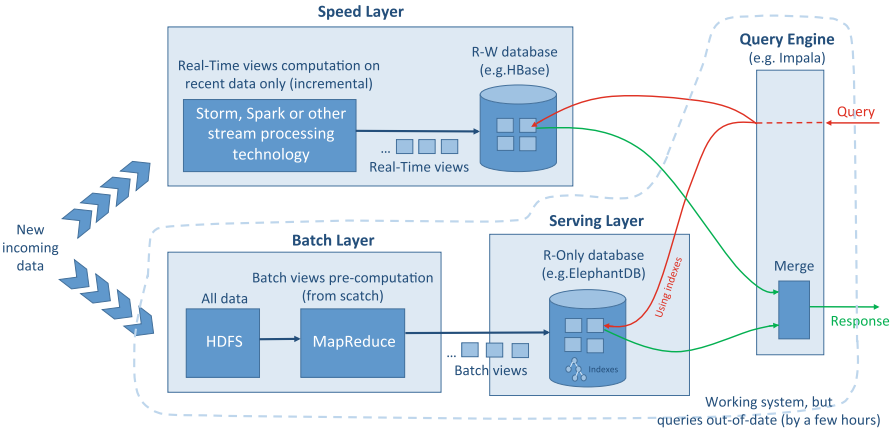


Fig. 4. Lambda architecture

analytics and real-time querying of streaming data. An obvious advantage of the Hadoop framework is its high performance with data analytics (write once and read many paradigm). On the other hand, E^2mC requires micro-services that are based on the (quasi) real-time availability of information, especially for Early Warning. The Lambda Architecture allows the project to benefit from Hadoop performance, while providing the infrastructure for the real-time processing of streaming data from social media.

4 Concluding Remarks

In this paper, we introduced the main requirements and discussed architectural choices for additional services for Emergency Management Systems, based on social media and crowdsourcing. Several issues are still open and need further investigation in the project. We mention here two important ones which are critical for the development of the project. First, as mentioned above user involvement is planned, assigning users tasks for obtaining further information and human analysis of the data. Such information, as discussed above, is inherently of low quality. In addition, it is important to identify possible groups of reliable and available users to perform the assigned tasks, to devise strategies and workflows to assign tasks and collect and integrate information, and in general creating communities of supporting users during the emergency phases. Another important aspect is the identification of the interested stakeholders for the use of the results of the project, since many different roles and institutions are usually involved in emergency response. It is important to identify a technical solution that is providing also an adequate support to the management of the system and involvement of several actors. The key roles and owners of the involved processes need to be identified and supported by an adequate distributed architecture, both in technical terms, and in ownership and responsibility.

Acknowledgments. This work has been developed in preparation for the E^{2}mC H2020 European project "Evolution of Emergency Copernicus services". This work expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work. The authors would like to thank the E^{2}mC project partners for their collaboration in setting up the project proposal.

References

1. Copernicus, a European system for monitoring the Earth. http://www.copernicus.eu/
2. E^{2}mC - Evolution of Emergency Copernicus services, Project H2020-EO-2016 (2016). http://www.e2mc-project.eu/
3. Flamini, A., Grandoni, D., Britti, F., Salvi, F.: e-GEOS capabilities in rapid emergency response. Two case studies: L'Aquila earthquake and Parma/Pepang typhoon. ISPRS Archive, vol. XXXVIII, Part 4-8-2-W9 (2010)
4. Francalanci, C., Giacomazzi, P.: TORCIA: una piattaforma collaborativa per la gestione delle emergenze, Mondo Digitale, no. 57 (2015)
5. Francalanci, C., Giacomazzi, P.: TORCIA - A decision-support collaborative platform for emergency management. In: DATE Conference (2015)
6. Lindsay, B.R.: Social Media and Disasters: Current Uses, Future Options, and Policy Considerations. Congressional Research Service 7–5700 (2011). www.crs.gov, R41987
7. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications Co., Greenwich (2015)
8. Meesters, K., van Beek, L., Van de Walle, B.:#Help. The reality of social media use in crisis response: lessons from a realistic crisis exercise. In: 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, pp. 116–125 (2016)

Challanges in Services Research: A Software Architecture Perspective

Flavio De Paoli(✉)

University of Milano-Bicocca, Milan, Italy
depaoli@disco.unimib.it

Abstract. Cloud computing and Internet of Things are imposing a dramatic change in software development and delivery. Moreover, ICT solutions are paving innovation in every sector and therefore becoming a business factor for the success of any enterprise. The implications are manifold since technical issues need to be harmonised with social, organisational and legal aspects. In this paper we illustrate and comment the current trends to identify research directions to build services as comprehensive components accessible via APIs. The goal is to deliver services ecosystems, which call for open platforms to manage services that can connect and interact via shared protocols in dynamic heterogeneous contexts. Machine-readable semantic descriptions, microservices (single-function services), and containers (independent units of deployment) are discussed as building blocks for software architectures of the future.

1 Introduction

In the last decades we dreamed software development by assembling software components. The dream may become true thanks to microservices[1], a new generation of services, which is likely to affect the ICT industry for the next decades. Microservices [10] can be defined as single-function components that can be composed and orchestrated to shape applications according to specific requirements (user needs and context). Microservices are of growing relevance due to the Internet of Things (IoT) era we are entering: sensors, transmitters, actuators and other electronic devices are able to collect and exchange data to populate the platforms of the future. We are taking part in the *servitization* of the real world.

Microservices will be the building blocks of a new generation of software that will deliver personalised solutions to end users and enterprises. The result is that computing is moving from data centres (that will keep their primary role of collecting, storing and analyse data) to the edge of the network where new processes (i.e., applications) will be developed involving both machines and people. The software architecture perspective needs to move from products to platforms that provide developers with the necessary flexibility to support the quick changes required by a highly dynamic market.

[1] http://www.reactivemanifesto.org/.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 219–227, 2018.
https://doi.org/10.1007/978-3-319-72125-5_18

This days, a popular example in the retail industry is to build applications that involve customers with personalised experience when they visit a physical shop. When a customer enter a shop holding a smartphone he or she becomes part of an *ecosystem* that includes sensors installed in the shop to collect real-time data, services accessing data from the retailer, and Internet services to access public data. Already from this simple example we can see that data as well as communication issues are becoming central. We need to rethink how services are build around data so to facilitate data exchange, retrieval and integration. The communication infrastructure has to provide a common platform to let services interact. A key characteristic of future applications will be the mobility, which calls for simple models and standards to be able to accommodate heterogeneous components (services interfacing devices, sensors, applications and people) developed by several stakeholders in different domains and with different purposes.

The challenge is to design and provide open platforms in which services can move, understand the new context, interact one each other and with the environment, accomplish a given task with the resources available in a specific context as well as in the Internet. Communication and management should follow a shared approach. The current REST-style API approach, which relay in the Web infrastructure, is a promising direction to provide lightweight interoperability and integration mechanisms.

2 An IoT Perspective

IoT is an emerging scenario for a future in which each "Thing" will be provided with a unique identifier and connected to transfer data over a network without requiring human-to-human or human-to-computer interaction. Things will be part of ecosystems that include computers and humans, but will be as much automated as possible. In practice, IoT provides connections between the physical world to the virtual world. IoT is mostly in charge of providing information to computers and humans to support their informed decisions (by using data collected from sensors), and tools to implement the decisions taken (by operating software and devices).

An emerging approach is to access and operate Things *as a Service*, according to the Service Oriented Architecture (SOA) approach, so to make them compatible with existing service-based systems [6]. IoT services differ from traditional software services for a number of issues, ranging from technical (scalability, heterogeneity) to social (security, privacy, trust)[2]. The main differences can be identified in the reduced capacities of devices involved in IoT, and in the continuous nature of data provided by sensor devices. As a matter of facts, resource, bandwidth and energy restrictions of the target devices are major limitations to adopt the REST/HTTP model to implement services associated with Things. The reason is the overhead required by HTTP messages, which is too

[2] See for example the final report of the Workshop on "next-generation Internet of Things": http://anrg.usc.edu/ngiot16.

heavy for simple devices. Moreover, HTTP is more suitable for discrete services based on request/response conversations, than for managing continuous data streams such as the ones coming from sensors.

There already exist proposals to deliver IoT-specific protocols that cope with the nature of involved Things. MQTT [1] and CoAP [12] are two reference standards at the moment. MQTT implements a publish/subscribe model, while CoAP follows the REST approach[3], which dominates the Web today. Performance evaluations of the two protocols can be found in [5,13], where the conclusions are that performance is highly dependent on the environment, which allow us to conclude that the characteristics of exchanged data are relevant for both efficiency and effectiveness.

The increasing size of the data managed by IoT-based systems has opened a debate on where the computation should occur. The two options are moving the computation *into the cloud* (i.e., at the data centres), or performing it *at the edge* of the network (i.e., at or near devices). The former has the advantage of virtually unlimited storage/computing power, and the disadvantage of moving massive amount of data over the network. The latter has the advantage of reduced traffic on the network, and the disadvantage of limited storage/computing power. The current trend is favouring edge computing since the continuous increment of the network traffic is unsustainable (due to bandwidth and latency), and processing every time a full dataset is often unneeded (a smaller set of relevant data allows for more effective answers to real-time requests). Moreover, effective personalisation, confidentiality, and timely reactivity call for application logic near the users (either humans or machines). As a consequence, devices will be equipped with gateways that provide aggregated views of data flows to near components that can perform real-time computing to promptly detect and react to changes. However, such data should be send to the cloud for further analysis, possibly off-line. The term *fog computing* was introduced to name systems that involve edge computing as counterpart of cloud computing [2].

In this scenario, microservices can play a role since they can be hosted at the edge of the network to implement specific tasks to interface devices and the rest of the applications. They can provide the right flexibility to ensure the evolution of applications. Open issues are the definition of (re)configuration mechanisms to make systems react to dynamic changes, and the level of protection that can be ensured to open systems in which components can join and leave arbitrarily. In the retail example sketched in the introduction, part of the application hosted by the shop could talk to the apps installed on the smartphone hold by the customer. At both sides software components and sensors are involved to connect and exchange data. Moreover, they should interact with the *cloud* to collect data (e.g., *insights*: weather conditions, related events, and user profiles), and provide fresh information on the current context (e.g., data collected by sensors installed in the shop, and information on customer behaviours).

[3] CoAP redesigns a subset of the HTTP methods to minimise the overhead, and extends the GET method to support basic publish/subscribe interactions, which is achieved by adopting the *observer pattern* to monitor state changes in a device.

3 Services Ecosystem

Since the beginning, services have been conceived as Lego blocks that can be freely composed to form arbitrarily complex systems and applications. The reality was made of systems built by manually adapting existing services, when not by developing services as part of a systems that cannot be, or can hardly be, used in other contexts. The emerging reality of IoT-based systems (that can take the form of smart city, smart home, smart mobility, ...) is pushing in the direction of investing to develop models, tools, an infrastructures to support truely modular services. The goal is the development of the so called *services ecosystems* [3]. We already introduced the idea of developing microservices as specialised actors of such ecosystems to implement single features/functions that can be composed with others to deliver an application. The separation-of-concern principle that sits behind the (empirical) theory of microservices let designers and developers concentrate on some well-defined aspect, and lead to loosely coupled services that can be developed and deployed independently.

Besides a set of explicit benefits that include increased agility, resilience, scalability, and developer productivity, there is a set of *hidden dividends of microservices that implementers should make a conscious effort to reap* [7]. A detailed discussion of such benefits is out of the scope of this paper, but, it is worth mentioning that there are relevant organisational aspects that make a change in the software development philosophy: a single multidisciplinary team is in charge of the whole service life cycle, including design, development, deploy and maintenance. A major effect is the end of centralised data centres as unique model to store and analyse data. Which is what IoT systems call for: distribute storage and data processing to reduce latency and enhance protection.

As discussed in the next section, the use of containers to host microservices along with dependencies supports the creation of components that are independent units of management and distribution. If we adopt a REST-like approach to provide them with communication and interaction capabilities, we can get components that are very close to the Lego blocks envisaged in the early stages of SOA. REST-like standard interfaces enables for automatic management and facilitate composition of components.

From an architectural point of view, we can distinguish three types of services: front-end services, which form the *systems of engagement*[4], and the back-end services that can be further refined in cloud services, which form the *systems of records*, and may include the *systems of insight* to further analyse data to provide advises. We need to say that a shared consensus on such a unified architectural organisation is still under discussion in the different domains and that a general agreement is far to be reached. The overall goal is to identify the roles of components in a multi-tier architecture. Anyway, some elements are consolidating: the evidence that the cloud cannot be the only repository for data and the only execution environment for applications since businesses need to stress their

[4] The term *Systems of Engagement* was introduced by Geoffrey Moore in a white paper entitled: "Systems of Engagement and the Future of Enterprise IT" [9].

peculiarities and protect their own identity, and the evidence that new devices enable for new solutions to *engage* customers, partners and employees.

The issue is well represented by the shop examples already introduced: smartphones, as user-agent systems, need to engage services in a given context. They act as actors in a service ecosystem that need to *discover* the available services, *select* the ones that fit their requirements, and finally *compose* their behaviour to deliver new comprehensive systems. We are using the terminology introduced with SOA to emphasis that the solutions devised in the past do not fit the dynamics of new contexts. In the next sections, we discuss three of the most affecting challenges we need to take up.

4 Programmable Platforms

Even if the de-facto reference infrastructure is the Internet for basic communication (TCP/IP and UDP/IP are the transport protocols) and the Web to support services interaction (HTTP is the application protocol for API development), there is still little convergence on the best way to develop services as node of a distributed application. Among others, open challenges are to cope with continuous integration and continuous delivery that have become a common mantra for the platform of the future; security and privacy protection that are increasingly important in an open world that foresees application, device and network convergence; and finally, management (operation and dynamic configuration) of multi-tenant heterogeneous systems.

To support such scenarios, we need to provide open programmable platforms that can ensure flexibility in services management to address adaptability to rapid change and support evolution of both participating services and features extension. In other terms, we need to provide microservices, which implement the needed functions, with hosting platforms that provides suitable execution and management support. Moreover, platforms should be able to address cross-cutting aspects to let microservices implement functionalities with the expected qualities. A possible direction is to exploit *containers* as programmable platform components that can be configured and managed in heterogenous and distributed environments[5].

A container can be defined as a sandbox that can (i) host an application along with all the required software to run, and (ii) be deployed in an host operating systems. These two characteristics make containers a perfect home for microservices. Once developed, a microservice is compiled to deliver an *image* that becomes a component that can be executed in a container, stored, retrieved, and sent over the network. Being standardised units for software development, containers facilitate the life-cycle management of microservices to achieve, for example, scalability and availability. Compared to other solutions, such as virtual machines, containers do not have the high overhead and hence enable more efficient usage of the underlying system and resources.

[5] Docker (https://www.docker.com) is the most popular container technology.

Although promising, container technology is still young, and open issues need to be addressed. The major problem deals with security. Virtual machines, a mature and popular technology for deploying software, run applications inside a guest Operating System that ensure full process isolation. Instead, containers rely on the low-level mechanics of the host operating system to provide most of the isolation of virtual machines at a fraction of the computing power. Simplicity and performance come at a cost of lower standard control over the execution.

5 Services Description

To be part of an ecosystem, a service need to advertise its characteristics to let other services understand what is the communication style (e.g., by REST-style APIs), how the interaction may occur (e.g., order and semantics of exchanged messages), what kinds of data are exchanged (e.g., types of exchanged data), and finally what properties are guarantied (e.g., latency, scalability, security, costs).

In the past decade a vast literature has been published on service descriptions, in particular adopting Semantic Web approaches, since a major (or possibly the major) concern is to let machines *understand* and *infer* from description contents. Primarily, machines need to understand whether data produced by a service are of the same type requested by another service, or understand if they are compatible and may be transformed to become compliant, thus enabling services compositions. Despite of the huge effort none of the proposed solutions proved to be effective. The evidence is that available descriptions are largely incomplete and written in natural language (e.g., the ones in ProgrammableWeb, the current reference service directory[6]). Most of description languages use *name*:*value* pairs to describe features with no semantics associated with such *name* and *value* literals/strings, or with semantics written in natural language that only humans can understand. That's the case of Swagger[7], RAML[8], and API Blueprint[9], which are the most popular specification languages.

The use of REST as unified model to design interfaces and interaction between services has radically changed the approach to services design by moving from *function* oriented to *resource* oriented models. Therefore, services descriptions should concentrate on the exchanged data (i.e., the representations of the resources), and on the properties associated with a given service, the *non-functional properties* (NFPs) that qualify a service. NFPs include properties of exchanged data (e.g., the bit rate of a mp3 file), quality of services (QoS) (e.g., latency), and service level agreements (SLA) (e.g., legal clauses on usage). NFPs are often collected in *contacts* that should be described in a machine-readable formats that enable property matching [4].

[6] http://www.programmableweb.com/.
[7] https://www.swagger.io. Now Open API specification by OAI (Open API Initiative) specification http://openapis.org/.
[8] http://raml.org/.
[9] https://apiblueprint.org/.

We believe that an evolution of the Linked Open Data (LOD) approach in connection with the REST-style resource management can make descriptions an integral part of the Web, which means delivering descriptions as resources in machine-readable formats that can be accessed and managed with REST APIs. A path can be annotating descriptions by adding links to shared vocabularies/ontology that machine can exploit for automatic discovery, selection and composition. Moreover, we seek methods and tools to support (semi)automatic profiling of services to facilitate the construction of descriptions. We can take advantage of already available vocabularies (e.g., Linked Open Vocabularies [14]) to set up a solid shared knowledge, and adopt standard annotation guidelines (e.g., Dublin Core Application Profile (DCAP) guidelines[10] to share data semantics in a shared representation format. Examples in that direction can be found in [8,11].

6 Services Composition

Services composition is likely to be the most intriguing and challenging issue, and, despite the huge effort in the past years, the most unexplored. A reason may be found in the heterogeneity of descriptions, interfaces and modes of access/interaction that led to specific solutions instead of building shared models. As already discussed, the progressive convergence towards shared models (Internet and Web as infrastructure) can define a favourable framework for the development of general composition techniques.

A *composition* can be defined as a process that involve a set of services in a given workflow, or a self-organising system where services collaborate to reach a common goal. The former was traditionally classified as *orchestration* and was deeply investigated in terms of business processes, and delivered languages like the well-known Business Process Modelling Language (BPML). The latter was classified as *choreography* and was mainly investigated in agent-based systems, and much less in service-oriented systems.

More recently, end-user composition has been addressed by tools that support the creation of simple workflows that connect two or more services[11]. The common approach is to build *connectors* that enable inter-service communication. The result is a set of close and incompatible environments, which means that every component must conform a set of given rules to participate. The challenge is to develop open systems. The extensive use of REST APIs, in combination with semantic descriptions and implemented by microservices and containers may offer this opportunity.

Services can be consumed either by humans or by machines. The design of open platforms can provide support for composition (orchestration/choreography) by machines, but the real challenge is to invent new

[10] http://dublincore.org/documents/profile-guidelines/.

[11] For example IFTTT (https://ifttt.com/discover), Ziper (https://zapier.com/), Microsoft Flow (https://flow.microsoft.com) and IBM WebSphere Cast Iron Cloud integration (https://www.ibm.com/support/knowledgecenter/SSGR73).

metaphors and new environments to let humans understand and interact with new *ecosystems* of services. In the smart shop example, this means that customers should be able to understand, select, compose and orchestrate the services available in that specific context though the smartphones they are using.

7 Conclusions

The progressive convergence toward the Web as standard communication environment opens new perspectives to open software architecture developments to host services ecosystems. The use of (i) semantic descriptions allows for the definition of automatic matching mechanisms; (ii) REST-style APIs defines standard inter-service interaction; and (iii) containers facilitates services management.

References

1. Banks, A., Gupta, R. (eds.): MQTT version 3.1.1. Technical report, OASIS (2014). http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
2. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: 1st MCC Workshop. ACM, Ambleside (2012)
3. Bosch, J.: From software product lines to software ecosystems. In: Proceedings of the 13th SPLC 2009, pp. 111–119. Carnegie Mellon University, Pittsburgh (2009). http://dl.acm.org/citation.cfm?id=1753235.1753251
4. Comerio, M., Truong, H.-L., De Paoli, F., Dustdar, S.: Evaluating contract compatibility for service composition in the $SeCO_2$ framework. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC/ServiceWave-2009. LNCS, vol. 5900, pp. 221–236. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10383-4_15
5. Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., Katos, V.: Which IoT protocol? Comparing standardized approaches over a common M2M application. In: Proceedings of IEEE GLOBECOM 2016. IEEE, December 2016
6. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In: Internet of Things (IoT) 2010, pp. 1–8, November 2010
7. Killalea, T.: The hidden dividends of microservices. Queue **14**(3), 10:25–10:34 (2016). http://doi.acm.org/10.1145/2956641.2956643
8. Lucky, M.N., Cremaschi, M., Lodigiani, B., Menolascina, A., De Paoli, F.: Enriching API descriptions by adding API profiles through semantic annotation. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 780–794. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46295-0_55
9. Moore, G.: Systems of engagement and the future of enterprise it. Technical report, AIIM, Silver Spring (2011)
10. Newman, S.: Building Microservices, 1st edn. O'Reilly Media Inc., Sebastopol (2015)
11. Panziera, L., De Paoli, F.: A framework for self-descriptive restful services. In: Proceedings of the 22nd WWW Conference Companion, pp. 1407–1414 (2013)
12. Shelby, Z., Hartke, K., Bormann, C.: The constrained application protocol (CoAP). Technical report, IETF (2014). http://dx.doi.org/10.17487/RFC7252

13. Sheng, Z., Wang, H., Yin, C., Hu, X., Yang, S., Leung, V.C.M.: Lightweight management of resource-constrained sensor devices in Internet of Things. IEEE Internet Things J. **2**(5), 402–411 (2015)
14. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked open vocabularies (LOV): a gateway to reusable semantic vocabularies on the web. Semant. Web J. **1**, 1–5 (2014)

PhD

Preface of PhD Symposium

The goal of the PhD-Symposium is to provide a forum for PhD students to present and to discuss their work with senior scientists and other PhD students working on related topics. As for the main conference, the topics focus on all aspects of Cloud Computing, Service Oriented Architectures, Web Services, and related fields. In contrast to the main conference, this work is usually unfinished or has just been started in the PhD projects. The programme committee carefully selected five contributions. Each submission was reviewed by at least two PC-members. In addition to the precise description of the problem to be solved, preliminary results, and first ideas for solving the main problem, the contributions also include a workplan. All these issues have been discussed at the symposium with selected senior scientist and the PhD students. After the symposium, the students who presented results already mature for a scientific publication, have been invited to prepare a paper presenting and discussing them. This post-symposium proceedings includes such papers.

We are grateful to the conference organizer Schahram Dustdar and his team for his organizatorial support.

Organization

PhD Symposium Organizers

Gianluigi Zavattaro	University of Bologna, Italy
Wolf Zimmermann	University of Halle, Germany

PhD Programme Committee

Antonio Brogi	University of Pisa, Italy
Friederike Klan	Friedrich Schiller University Jena, Germany
Welf Löwe	Linnaeus University, Sweden
Flavio de Paoli	University of Milano-Bicocca, Italy
Alexander Pokahr	University of Hamburg, Germany
Ernesto Pimentel	University of Malaga, Spain
Emilio Tuosto	University of Leicester, UK
Massimo Villari	University of Messina, Italy
John Erik Wittern	IBM T. J. Watson Research Center, USA

Towards a Unified Management of Applications on Heterogeneous Clouds

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

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

Abstract. The diversity in the way cloud providers offer their services, give their SLAs, present their QoS, or support different technologies, makes very difficult the portability and interoperability of cloud applications, and favours the well-known vendor lock-in problem. We propose a model to describe cloud applications and the required resources in an agnostic, and providers- and resources-independent way, in which individual application modules, and entire applications, may be re-deployed using different services without modification. To support this model, and after the proposal of a variety of cross-cloud application management tools by different authors, we propose going one step further in the unification of cloud services with a management approach in which IaaS and PaaS services are integrated into a unified interface. We provide support for deploying applications whose components are distributed on different cloud providers, indistinctly using IaaS and PaaS services.

Keywords: Cloud applications · Multi-deployment · TOSCA Brooklyn

1 Introduction

In recent years, Cloud Computing [1] has experienced a growth in the demand of its services. The Cloud promotes on-demand access to a large number of resources throughout three service models, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [2], which allow cloud providers to offer services for current IT requirements, with scalability and elasticity as the most relevant ones, and allow users to tailor the used resources to their needs.

Vendors such as Google, Amazon, Cloud Foundry, etc., have implemented their solutions to this model by developing their own cloud service layers, with custom APIs that expose their resources. Most of these providers offer a set

This work has been partially supported by Spanish MINECO/FEDER projects TIN2014-52034-R and TIN2015-67083-R; Andalusian Gov. project P11-TIC-7659; and Univ. Málaga, Campus de Excelencia Internacional Andalucía Tech.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 233–246, 2018.
https://doi.org/10.1007/978-3-319-72125-5_19

of similar services as regards functionality, but developed according to their own specifications. E.g., each supplier specifies its own Service Level Agreement (SLA) or Quality of Service (QoS), supports a concrete set of technologies, etc. The proliferation of these solutions has also increased the number of issues to be addressed in cloud computing, mainly related to the diversity of providers and their solutions, giving place to the vendor lock-in problem [3], and hampering the portability and interoperability in the definition and usage of services.

Due to this lack of standardisation, developers are often locked-in to concrete cloud environments, since they have to adapt their developments according to the specifics of the vendors that will be used to run their applications. This heterogeneity affects the entire lifecycle of systems, from design time to release/deployment, which complicates the development of portable applications and the integration of services of different providers to achieve cross-deployments. In this context, migrating components between different platforms seems impossible.

Given the current state of Cloud Computing, it looks reasonable to offer to developers mechanisms to deal with the restrictions to the portability and interoperability of applications. From the developers' point of view, we believe it would be very useful to have an environment in which we could build full detailed application descriptions in an agnostic way, supporting the use of services of different offerings to deploy our applications, and abstracting from the constraints of concrete providers. Furthermore, it would make sense to distribute the different modules of an application over services of different providers. This would allow us to optimise the usage of cloud resources, since we could select, for their deployment, and given the requirements of each of the modules of an application, and requirements of the application itself as a whole, the services with best features for each of the modules of our application. Moreover, we plan to go one step further, and analyse the portability between abstraction levels, initially focusing on IaaS and PaaS.

Once applications have been deployed and are running, using services of specific providers, developers may need to modify the cloud environment where the application is being executed due to many reasons, such as an application updating or different cloud events. For example, the performance of services could be altered, e.g., by a modification in the QoS by the provider, affecting the application performance or its cost. Developers could also modify the cloud resources used by their applications, for example, by adding new cloud services to provide new application features. It may be useful counting with mechanisms supporting the management and reconfiguration of cloud applications.

Additionally, it may be useful to users to have facilities for the migration of application modules between different cloud levels in order to maintain the performance and optimise the resources usage and minimise the cost. For instance, given an increase in the workload of an application, it could be beneficial to migrate some of its modules to PaaS, in order to take advantage of the automatic scalability facilities of this kind of services.

The rest of the paper is structured as follows. Section 2 describes our research challenges. The research plan and the current state of our research are explained in Sects. 3 and 4, respectively. Section 5 describes the current state of the prototype being under development. Finally, Sect. 6 presents our conclusions and future work.

2 Research Challenges

The main goal of this our wotj is to develop an environment that offers an homogeneous management of IaaS and PaaS services, and enables a methodology to describe applications and the required target cloud resources, providing developers with mechanisms to improve the portability and interoperability of applications. Moreover, it will allow users to choose the cloud resources whose features best adapt to their applications' requirements, with support for the deployment of each of the application modules using the PaaS or IaaS services that better fit their needs. In the following, we elaborate on the descriptions of our goals.

Unification of IaaS and PaaS cloud services. We plan to develop a common API that will unify cloud services independently of their abstraction level, for IaaS and PaaS. To achieve this, we will analyse the different service features and restrictions in order to find common patterns and abstract them under a unified interface. This unified level will offer to users a transparent and simple usage of different cloud services, allowing them to focus on the functionality of these services, while the complexity of using and integrating their interfaces is hidden by the unified API. We plan to build this API by homogenising services with different properties in order to build a normalised upper layer. Given the existing diversity, trying to homogenise all the functionalities of each provider will most probably not be possible. To minimise this problem, we will try to maintain these functionalities by using lower layers, with the goal of providing as many services as possible.

Description of applications and cloud services. We believe that the way to address the portability and interoperability issues is by developing an agnostic modeling framework to describe applications and the used cloud (IaaS and PaaS) services and resources. With this framework, users will be able to build full-detailed descriptions of their applications, including all the knowledge about the capabilities, requirements, kinds of services to run the application, etc., regardless of the concrete providers over which the application will be finally deployed. We plan to build on current standards, such as CAMP and TOSCA, in order to propose a standardised, powerful and flexible application-modeling environment.

Integration of the modeling and the unified API. A unified API will offer a homogeneous management of different services. An application model will allow us to detail all the knowledge about an application. Then, we believe that by joining both elements, API and agnostic modeling, we will be able

to provide an environment which will allow portable applications to be modeled and deployed using the unified API features in a standardised manner, providing a complete application lifecycle management. Then, any services supported by the unified API will be available for users to deploy modeled applications without requiring any knowledge about the concrete provider interfaces.

Development of a functional prototype. We will develop a functional prototype in which we will experiment with the accomplishments related to the previous goals, and to show its viability and to evaluate its advantages and disadvantages.

Post-deploy management of applications. Although not one of the core goals of the work being described, we will also study the implications of our proposal on the management of applications once they have been deployed and are running. Specifically, we will consider aspects such as the monitoring of cloud applications whose modules are deployed using services of different providers, possibly at different levels, and how SLA policies may be specified (e.g., auto-scaling policies).

Hot reconfiguration of applications. Given agnostic application descriptions, it seems natural to consider the possibility of moving application modules from the services they are deployed on to other ones with better features, or for a better adjustment of the application needs. We will consider the possibility of performing such reconfiguration operations at runtime.

3 Research Agenda

In this section we describe the different phases in which we have structured our work plan, detailing the tasks for each of them.

Analysis of the Related Work

- Exhaustive analysis of the state of the art on homogenisation and cloud management. We will review current practical and theoretical proposals and related standards. We will also analyse their implementation plan.
- Systematic analysis of the features and restrictions of the different cloud offerings in order to determine the key aspects to consider when carrying out the proposed homogenisation. This will be made by defining different deployment use cases involving different service levels.
- Study of deployment-related concepts using services of multiple clouds (multi-clouds).
- Review of related open projects, with special emphasis on those using standards, including an evaluation of their capabilities and limitations.

API Composition and Unification of IaaS and PaaS Services

- Classification of different cloud services in terms of their functionalities and the services of the cloud offerings that will be supported by our approach, establishing a preliminary approach of the unified API.

- A first prototypical development of the unified API. We will most probably first develop independent versions for IaaS and PaaS, which will later be unified under a common interface.
- Our implementation efforts will be integrated inside an existent open project supported by an active community. We will pay special attention to Apache Brooklyn[1], an open project that offers a flexible and robust management of IaaS services of a large number of providers.

Application Modeling

- Analysis of the different concepts related to the management of applications and cloud services that will be supported by our modeling facilities to provide flexible and extensible mechanisms to describe systems.
- Development of a modeling proposal, supporting the definition of applications according to the results of the previous step, addressing the significant management and capabilities differences between the different providers. We will also study the use of current standards, initiatives and open projects focused on the normalisation of applications and the description cloud services.
- Development of a generic nomenclature to identify and reach the target providers that will be used to deploy applications, making sure that the nomenclature is flexible enough to support as many as possible provider properties, and enabling the distribution of the different application modules over different providers (cross-deployment).

Validation of the Proposal

- Revise the diversity of use cases proposed on the first phase focusing on different characteristics in order to check the supported providers under diverse restrictions.
- Application of the use cases to specific deployment scenarios which will be composed by different providers according to real situations.

Post-deployment Strategies

- As possible extensions, we will consider the monitoring concepts and mechanisms to add them to the common API and the application modeling.
- We will research on management policies, such as auto-scaling, which will be based on the previous monitoring experiments.
- We will study migration techniques, determining how application modules can be moved between services of different providers and its abstraction levels.

4 Current State of Our Proposal

We present in this section some of the goals we have already achieved.

[1] Apache Brooklyn: https://brooklyn.apache.org/.

4.1 Application Modeling

There is a lot of work on methodology descriptions in the literature, including many projects, standards and initiatives, as Cloud4Soa [4] CAMP,[2] Roboconf,[3] and mOSAIC.[4] After analysing the most relevant related work, we consider TOSCA (Topology and Orchestration Specification for Cloud Applications)[5] as a standard that provides a useful framework on which basing our application modeling. It defines a very flexible model for the description of cloud applications, the corresponding services, allowing their relations to be specify explicitly by using a fully service topology, containing all the knowledge about the applications. Furthermore, it allows the description of procedures to manage services using orchestration processes by using plans.

Currently, we only take advantage of the topology specification of TOSCA, what allows us to describe the knowledge about applications independently of any cloud resource restrictions, and integrate the different features and requirements of the different provider abstraction levels in the same model.

4.2 Towards a Unified API

We propose the development of a common API to unify the management of IaaS and PaaS services. After analysing the mechanisms to manage the cloud of different alternatives, such as OpenTOSCA,[6] Cloudify,[7] Alien4Cloud,[8] Cloud4Soa and Brooklyn, we decided to base our work on Apache Brooklyn, an open project with an active community behind. Brooklyn can manage the provisioning and deployment of cloud applications, can monitor applications' health and metrics, and handle the dependencies between components. It enables cross-computing features through a unified API to manage IaaS services offered by various providers.

Thanks to jClouds,[9] Brooklyn provides an API for the management of IaaS cloud services for a great number of providers and establishes a lifecycle for the management of services and applications. We have extended this API with facilities for the management of PaaS services of platforms based on Cloud Foundry, providing an homogeneous access to IaaS and PaaS services [5]. We have integrated the PaaS management in all the Brooklyn levels but without modifying its API. Then, we have obtained a prototype with a common API that manages IaaS and some PaaS services (currently, Cloud Foundry-based platforms) in a unified manner. We have tested this API by building portable applications, and deploying them using different IaaS and PaaS providers. Indeed, we have obtained in this way a first implementation of the proposed trans-cloud mechanisms.

[2] CAMP Standard: https://www.oasis-open.org/committees/camp/.
[3] Robocobf: http://roboconf.net/.
[4] mOSAIC: http://www.mosaic-cloud.eu/.
[5] TOSCA: http://docs.oasis-open.org/tosca/TOSCA/v1.0/.
[6] OpenTosca: http://www.iaas.uni-stuttgart.de/OpenTOSCA/.
[7] Cloudify: http://getcloudify.org/.
[8] Alien4Cloud: http://alien4cloud.org/.
[9] jClouds: http://jclouds.apache.org/.

4.3 Trans-cloud Management

Independent tools and frameworks have emerged with the goal of integrating, under a single interface, the services of multiple public and private providers (see, e.g., [6,7]). In a very short time, these platforms have evolved according to the mode in which developers can take advantage of integrated cloud services to expose and run their systems. Terms such as multi-cloud [8], cross-cloud [9], federated clouds [10], or inter-clouds [11] have been used for deployment platforms with the ability of distributing modules of an application using services from different providers.

The main differences between these approaches lie on the different ways of handling the connections between modules deployed on different platforms. However, in all these attempts, platforms allow operating simultaneously with a single level of service to deploy applications, i.e., all the components of an application are deployed either at the IaaS level or all at the PaaS level (see, e.g., [4,9,12]). From this, with the goal of unifying cloud services, we propose a second dimension in which deployment tools integrate IaaS and PaaS levels under a single interface. Then, this will allow developers to deploy their applications combining services offered by providers at any of these levels. Following the evolution in terminology, *multi-/cross-/inter-cloud*, we envision *trans-cloud* management tools without the limitations we currently have. *Trans-cloud* mechanisms enable one of the most important goals of this work, the unification of IaaS and PaaS cloud services (see Sect. 1). The idea behind *trans-cloud* is to be able to build our applications by using available services and resources offered by different providers, at IaaS, PaaS or SaaS level, using virtual machines or containers, according to our needs and preferences. We will focus on IaaS and PaaS in this work.

A graphical representation of our solution for trans-cloud is depicted in Fig. 1. Given a TOSCA YAML description of the application's components and their relationships, i.e., its topology, (an extended version of) Apache Brooklyn is used to perform the deployment of the application. TOSCA YAML descriptions are processed by a TOSCA Engine that generates the necessary objects for Brooklyn, performing the request and management of the services required for the deployment of applications.

Developers can describe their applications by means of our TOSCA YAML-based profile application, thus modelling in an agnostic way, and avoiding any cloud resource or service particularities. Then, they can point to the providers where the different application's modules will be deployed and use the trans-cloud approach for their distribution, avoiding the need of expertise in the management of the different vendors.

Brooklyn does not offer native TOSCA support. Instead, we take advantage of the Brooklyn-TOSCA project[10] to build the TOSCA Engine Brooklyn, and add to Brooklyn the capacity for the management of TOSCA specifications. Brooklyn-TOSCA is an open project being mainly developed by CloudSoft

[10] Brooklyn-TOSCA: https://github.com/cloudsoft/brooklyn-tosca.

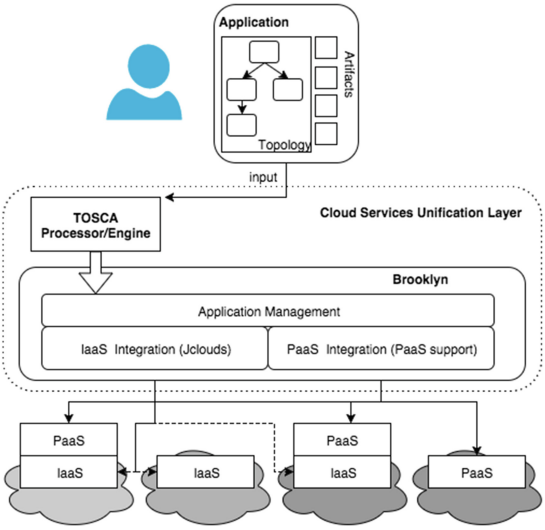


Fig. 1. Trans-cloud approach architecture

and FastConnect, which are the main developers of Brooklyn and Alien4Cloud, respectively. We have collaborated to this project with the goal of adding capacity for deploying and managing applications and cloud resources through TOSCA concepts. In the following section we explain our trans-cloud approach by means of an use case and some tentative proposals to improve our work.

5 The Softcare Case Study

In this section, we introduce a case study to illustrate how trans-cloud operates and how we propose using the TOSCA standards. We also give some high-level presentation of the implementation of our solution for trans-cloud.

The Softcare case study corresponds to an application for social inclusion of elderly people and for the management of their medical problems. The Softcare application was developed by Atos Spain in the context of the SeaClouds project [13–15]. Softcare is a cloud-based clinical, educational, and social application, based on state-of-the-art technology, that provides an innovative and integrated solution with the following main features: home as care-environment through the provision of user-friendly ICT tools for frequent, unobtrusive monitoring; risk assessment and early detection of deterioration symptoms; high-quality interaction between doctors, social workers and elderly people; monitoring and follow-up of the elderlies' progress; and self-care and self-management of chronic conditions, through the development of social networking, educational tools and nutrition recommendations.

Figure 2 shows the topology of our Softcare case study as defined using the Alien4Cloud TOSCA editor.[11] We can observe that the application is composed of five modules, each one modeled using a NodeTemplate: three web modules over respective Tomcat servers (notice the Tomcat icons) and two MySQL databases (notice the Database icons). The Softcare Dashboard component contains the main graphical user interface, which depends on Forum and SoftcareWS modules. Forum adds a forum service to the web platform, and SoftcareWS contains the application's business logic. Databases ForumDB and SoftcareDB are modeled by MySQL NodeTemplates. ForumDB stores the forum's messages, and SoftcareDB stores all the other data of the application.

Fig. 2. Softcare topology schema

Listing 1.1 shows the Softcare topology YAML schema using the proposed application modelling. Notice the ellipses in the YAML: we have removed detailed information on properties, capabilities, etc., with which we specify the locations of the war files to be deployed, ports to be used, and other details necessary for the correct operation of the application. This information has been removed to save some space, since it is not relevant to our presentation. NodeTemplate's requirements have been kept to show the dependency relations between components, according to the application topology (see Fig. 2). As we can see in Listing 1.1, we can difference two distinguish parts. First, we find the agnostic and portable application description that only contains information about the application components and their relations. In the second part, lines 43–51, information on target providers, following the Brooklyn-TOSCA[12] initiative, is provided. In this part, each topology component indicates to the provider where it will be deployed by means of TOSCA policies (*brooklyn.locations*) and groups. TOSCA groups allow grouping one or more node templates for assigning special attributes, like policies. Brooklyn-TOSCA takes advantage of policies flexibility to describe the providers where to distribute the application modules. In this case, we can see how two groups have been defined, for components to be deployed on Amazon (Oregon's Cluster), and SoftLayer (Seattle's cluster). In this case, both of them are IaaS services.

The separation of the topology and the providers descriptions allows us to ensure the independence between the application description and the used cloud resources, facilitating a better portability management. The application's topology can be specified at design time, without referring to cloud resources, and target providers can be supplied in very late stages of the design, at deployment phase. This is a clear advantage for developers and administrators of the

[11] Alien4Cloud editor: http://alien4cloud.github.io/.
[12] Brooklyn-TOSCA project https://github.com/cloudsoft/brooklyn-tosca.

```
tosca_definitions_version: tosca_simple_yaml_1_0_0_wd03

imports:
  - tosca-normative-types:1.0.0.wd06-SNAPSHOT
  - mysql-type:2.0.0-SNAPSHOT
  - tomcat-template:5.0.2-SNAPSHOT
...

description: SoftcareApp topology

topology_template:
  node_templates:
    SoftcareDashboard:
      type: org.apache.brooklyn.entity.webapp.tomcat.TomcatServer
        ...
      requirements:
        - endpoint_configuration:
            node: SoftcareWS
            ...
        - endpoint_configuration:
            node: Forum
            ...
    Forum:
      type: org.apache.brooklyn.entity.webapp.tomcat.TomcatServer
      ...
      requirements:
        - endpoint_configuration:
            node: ForumDB
            ...
    SoftcareWS:
      type: org.apache.brooklyn.entity.webapp.tomcat.TomcatServer
      ...
      requirements:
        - endpoint_configuration:
            node: SoftcareDB
            ...
    ForumDB:
      type: org.apache.brooklyn.entity.database.mysql.MySqlNode
      ...
    SoftcareDB:
      type: org.apache.brooklyn.entity.database.mysql.MySqlNode
      ...
  groups:
    add_compute_locations:
      members: [ SoftcareDB, Forum, ForumDB ]
      policies:
      - brooklyn.location: aws-ec2:us-west-2
    add_web_locations:
      members: [ SoftcareDashboard, SoftcareWS ]
      policies:
      - brooklyn.location: softlayer-seattle
```

Listing 1.1. Excerpt of Softcare's YAML TOSCA topology

```
tosca_definitions_version: tosca_simple_yaml_1_0_0_wd03

imports:
  - tosca-normative-types:1.0.0.wd06-SNAPSHOT
  - mysql-type:2.0.0-SNAPSHOT
  - tomcat-template:5.0.2-SNAPSHOT
...

description: SoftcareApp topology

topology_template:
  node_templates:
    ...
  groups:
    add_compute_locations:
      members: [ SoftcareDB, Forum, ForumDB ]
      policies:
      - brooklyn.location: aws-ec2:us-west-2
    add_web_locations:
      members: [ SoftcareDashboard, SoftcareWS ]
      policies:
      - brooklyn.location: pivotal-ws
```

Listing 1.2. Adding new locations to web modules

applications, since the deployment of applications becomes more flexible. For instance, we may decide to deploy some or all of the components in our Softcare case study on a different provider just by changing the corresponding location. Listing 1.2 shows a modified version of the excerpt of the YAML TOSCA in Listing 1.1, focusing on the groups section. Now, the group with the components SoftcareDB, Forum, and ForumDB will be deployed on the Oregon's cluster of AWS, IaaS provider, as before, while the group with the components SoftcareDashboard and SoftcareWS will be deployed on Pivotal Cloud Foundry as PaaS. Of course, modules could be grouped differently, and deployed at will on any IaaS or PaaS platform available in Apache Brooklyn or the PaaS support we have developed based on Cloud Foundry.

6 Conclusions and Future Work

We propose the development and use of a common API to unify the management of IaaS and PaaS cloud services, making their use completely uniform. We allocate this proposal inside what we call *trans-clouds*, which extends cross-cloud application deployment and management by supporting the portability and interoperability of application modules from different providers and at different levels. We propose a TOSCA-based agnostic modeling of applications and cloud services, which allows us to specify the characteristics and requirements of any system to be deployed in the cloud. The standardised description of applications and cloud resources and the homogenous service API significantly reduce

the portability and interoperability issues related to vendor lock-in, facilitating the reusability of cloud services. By having an agnostic model of our system may greatly simplify migration, or simply decision change. Indeed, with our approach, each component may be deployed at one level or the other just by changing its location. It is worth noting that the proposed thesis project is not an implementation exercise on an existing deployment tool, but an innovative general approach to ease the cloud deployment of applications, enforcing the independence of both cloud providers and cloud models.

We have developed an operational prototype built on the well-established Apache Brooklyn tool in order to test our trans-cloud ideas. Brooklyn provides support for a large number of IaaS providers. Thanks to our efforts in integrating Cloud Foundry into Brooklyn, it now also provides access to PaaS Cloud Foundry-based providers such as Pivotal Web Services or Bluemix.

Part of the research in this thesis was developed in the context of the Seaclouds project, and some preliminary results related to the thesis plan described here have already been published in [16–18].

Much work remains ahead. We plan to analyse new providers in order to extend the supported PaaS services and technologies. Our current model will be extended to integrate new PaaS providers, such as Heroku or OpenShift. Given their heterogeneity, the new providers to be considered will have to be carefully analyzed to find out how they should be added to our framework. Furthermore, we plan to study the possibility of using the flexibility and scalability mechanisms available for PaaS to develop management policies to react to applications' events.

To advance in the unification of PaaS and IaaS, we will explore the possibilities for improving the post-deployment facilities of our proposal for trans-cloud. Specifically, we will study the uniform monitoring of applications on both levels, and mechanisms for the migration of individual components of applications at runtime.

Given our unified API, to handle different vendors' cloud approaches, and a standards-based modelling of applications, the dynamic reconfiguration of applications seems to be a natural step to take. Once our trans-cloud tool receives an application description, it build an internal application model which contains all information about the application components, their relations and the used cloud services. Taking advantage of all this knowledge and the capability of unified API, once an application is running, we can move one or some of its components to different providers, to ensure specified restrictions, such as performance, cost, etc. For the definition of our migration mechanisms we will consider existing robust algorithms [19] for minimizing the degradation of performance during the process and maintaining the application topology by means of ensuring the inter-relations between components.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Youseff, L., Butrico, M., Silva, D.D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (GCE), pp. 1–10 (2008)
3. 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)
4. Zeginis, D., D'Andria, F., Bocconi, S., Gorronogoitia Cruz, J., Collell Martin, O., 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) (2013)
5. Carrasco, J., Cubo, J., Pimentel, E.: Bidimensional cross-cloud management with Brooklyn and Tosca. In: 2016 IEEE 9th International Conference on Cloud Computing. IEEE (2016)
6. Sellami, M., Yangui, S., Mohamed, M., Tata, S.: PaaS-independent provisioning and management of applications in the cloud. In: 6th International Conference on Cloud Computing (CLOUD), pp. 693–700. IEEE (2013)
7. Gonidis, F., Paraskakis, I., Simons, A.J.H.: A development framework enabling the design of service-based cloud applications. In: Ortiz, G., Tran, C. (eds.) ESOCC 2014. CCIS, vol. 508, pp. 139–152. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14886-1_14
8. 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)
9. Elkhatib, Y.: Defining cross-cloud systems. ArXiv e-prints, February 2016
10. 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)
11. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. Softw. Pract. Exp. **44**(3), 369–390 (2014)
12. Hossny, E., Khattab, S., Omara, F., Hassan, H.: A case study for deploying applications on heterogeneous PaaS platforms. In: International Conference on Cloud Computing and Big Data (CloudCom-Asia), pp. 246–253. IEEE Computer Society (2013)
13. Brogi, A., Carrasco, J., Cubo, J., D'Andria, F., Ibrahim, A., Pimentel, E., Soldani, J.: EU project seaclouds - adaptive management of service-based applications across multiple clouds. In: 4th International Conference on Cloud Computing and Services Science (CLOSER), pp. 758–763 (2014)
14. Barrientos, M., et al.: Adaptive application management over multiple clouds. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 422–424. Springer, Cham (2016)
15. 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)
16. Carrasco, J., Cubo, J., Pimentel, E., Durán, F.: Multi-deployment over heterogeneous clouds with TOSCA and CAMP. In: 6th International Conference on Cloud Computing and Services Science (CLOSER) (2016)

17. Carrasco, J., Cubo, J., Pimentel, E.: Towards a flexible deployment of multi-cloud applications based on TOSCA and CAMP. In: Ortiz, G., Tran, C. (eds.) ESOCC 2014. CCIS, vol. 508, pp. 278–286. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14886-1_26
18. Carrasco, J., Cubo, J., Durán, F., Pimentel, E.: Bidimensional cross-cloud management with TOSCA and Brooklyn. In: International Conference on Cloud Computing (CLOUD) (2016)
19. Durán, F., Salaün, G.: Robust reconfiguration of cloud applications. In: Proceedings of the 17th International ACM Sigsoft Symposium on Component-Based Software Engineering, CBSE 2014, pp. 179–184. ACM (2014)

Deadlock Analysis of Service-Oriented Systems with Recursion and Concurrency

Mandy Weißbach(✉)

Institute of Computer Science, Martin Luther University Halle-Wittenberg,
Von-Seckendorff-Platz 1, 06120 Halle, Germany
mandy.weissbach@informatik.uni-halle.de

Abstract. In this paper, we show an abstraction-based approach towards analysis of Service-Oriented Systems with the help of Process Rewrite Systems. On the one hand the approach takes into account recursion, i.e., internal recursion in service implementations as well as external recursion over service boundaries. On the other hand, also internal concurrency and concurrency over service boundaries are considered. The abstraction can automatically derived from the service implementations.

Keywords: Process rewrite systems · Deadlock · Recursion
Concurrency

1 Introduction and Motivation

Composition of services can be an error-prone task. In particular when for example web service descriptions only provide interface signatures such as WSDL-specifications. The lack of information might lead to unintended behaviour of services such as deadlocks, livelocks [27], unexpected abortions, unexpected functionalities. As interface signature do not provide enough information on the use of services, approaches such as semantic web technologies, tools for checking compatibility criterion etc. are used to reduce the risk of unintended behaviour due to composition incompatibilities [3,19,21].

The focus of this work is on deadlock analysis. Prominent approaches such as van der Aalst's workflow nets [23] are Petri-Net based and use Petri-Net tools to analyse deadlocks [24]. Other approaches are based on process-algebras [16] and use tools from this field to analyse deadlocks [20]. These approaches are usually refinement-based, i.e., the behaviour of a service is defined as a workflow net or process-algebraic expression and then refined to the service implementation. The behaviours are composed corresponding to the architecture of the service-oriented system and then e.g., checked for absence of deadlocks. In contrast to these works, we use an abstraction-based approach, i.e., the behaviour is abstracted from the service implementations. The motivation for an abstraction-based approach is that there are many services not developed according to a PRS-Rule refinement-based approach. Furthermore, even if they have been developed

Supervisor: Wolf Zimmermann.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 247–259, 2018.
https://doi.org/10.1007/978-3-319-72125-5_20

initially by a refinement-based approach, it is unlikely that programmers consistently maintain the implementation and its abstraction.

For abstraction-based approaches, it is necessary to automatically derive the abstract behaviour of a service from its implementation. Therefore, this abstraction is a classical translation task which can be implemented using compiler technology.

An abstraction-based approach should deal with all kinds of source programs. Following this philosophy, concepts like procedure calls, forking asynchronous procedure calls, synchronization and exception handling have to be considered, c.p. Table 2. However, Petri-Nets allow only rather imprecise abstractions of procedure calls. The reason is that the behaviour of recursive procedures corresponds to the LIFO principle and requires therefore a stack [15]. A solution for a more precise abstraction might be recursive Petri-Nets [11] or Mayr's process rewrite systems [18]. Recursive Petri Nets combine properties of Petri Nets and context free grammars. In [9] it is shown that recursive Petri Net languages strictly include the union of Petri Net and Context Free languages. The Process Rewrite System (PRS) are an extension of Petri-Nets by stacks [17]. Therefore, for modelling recursive procedures, forking asynchronous procedure calls, and synchronizations, Process Rewrite Systems and recursive Petri Nets are sufficient. Indeed, it was shown in [10] that PANs (a subset of the more expressive PRSs) are a subset of recursive Petri-Nets, but is unknown whether they are equal.

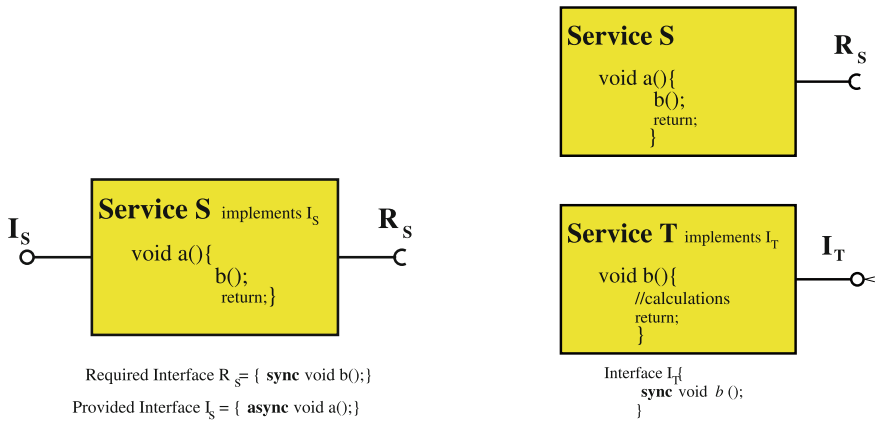
It was also shown that it is decidable to check PRSs for the absence of deadlocks, the deadlock reachability problem [18]. We are not aware of any work towards decidability of deadlock detection in recursive Perti Net. We therefore use PRSs as the basic model for specifying the abstract behaviour.

This paper is organized as follows: In Sects. 2.1–2.3 we introduce PRSs according to [17], service-oriented systems and we give a short overview of the abstraction process and discuss the deadlock issue. In Sect. 2.4 we present the problem by means of an example of a service-oriented system including procedure calls, forking asynchronous procedure calls and conditionals. We show how a service-oriented system can be described with Process Rewrite Systems. In Sect. 3 we discuss the one-to-one correspondence of process rewrite systems with cactus stacks and in Sect. 4 discusses related work and Sect. 6 concludes our work with a short overview of the gained results and gives a short outlook.

2 Foundation

2.1 Services and Service-Oriented Systems

We assume that a service S is an implementation with a provided interfaces I_s, where an interface is a set of procedure signatures. It is possible that a service calls procedures/functions of other services. The required interface R_s of S is the set of procedures of other services called by S, c.f. Fig. 1a. A service-oriented system is composed by two or more services which communicate over a required (and provided) interface, cf. Fig. 1b. Procedures of an interface can be either



(a) A Service S with provided interfaces I_S and the required interfaces R_S.

(b) A composition of Service S and Service T.

Fig. 1. Service S and a service-oriented system with two services.

called synchronous or asynchronous. Called synchronous procedures block their caller until the callee has been completed the call, while asynchronous procedure calls are executed in parallel when they are called. Hence the caller can proceed without waiting for completion.

2.2 Process Rewrite System

Mayr presented a unified view of Petri Nets and several simple process algebras by representing them as subclasses of the general rewriting formalism *Process Rewrite Systems* [18].

It is based on rewrite rules on process-algebraic expressions. The set $PEX(Q)$ of process-algebraic expressions over a finite set Q (*atomic processes*) is the smallest set satisfying:

(i) $Q \subseteq PEX(Q)$
(ii) If $e, e' \in PEX(Q)$, then $e.e' \in PEX(Q)$ and $e \parallel e' \in PEX(Q)$ (*sequential* and *parallel composition*, respectively).
(iii) $\varepsilon \in PEX(Q)$

The *empty process*, denoted by ε, is the identity w.r.t. sequential and parallel composition.

Definition 1 (Process Rewrite Systems). *A* Process Rewrite System *(short: PRS) is a tuple* $\Pi \triangleq (\Sigma, Q, \rightarrow, q_0, F)$ *where*

(i) Q *is a finite set (*atomic processes*),*
(ii) Σ *is a finite alphabet disjoint from* Q *(*actions*),*
(iii) $q_0 \in Q$ *(the* initial state*),*

(iv) $\rightarrow \subseteq PEX(Q) \times \Sigma \times PEX(Q)$ *is a set of* process-rewrite rules,
(v) $F \subseteq Q \cup \{\varepsilon\}$ *(the set of* final processes*).*

The PRS Π *defines a derivation relation* $\Rightarrow \subseteq PEX(Q) \times \Sigma^* \times PEX(Q)$ *(*Σ^* *is the set of all finite words over* Σ*) by the inference rules in Fig. 2.*

$$\frac{q \rightarrow q'}{q \Rightarrow q'} \text{ (R)} \qquad \frac{u \Rightarrow v}{u.w \Rightarrow v.w} \text{ (S)} \qquad \frac{u \Rightarrow v \quad v \Rightarrow w}{u \Rightarrow w} \text{ (T)}$$

$$\frac{u \Rightarrow v}{u \parallel w \Rightarrow v \parallel w} \text{ (P1)} \qquad \frac{u \Rightarrow v}{w \parallel u \Rightarrow w \parallel v} \text{ (P2)} \qquad \frac{}{u \Rightarrow u} \text{ (L)}$$

$u, v, w \in PEX(Q)$

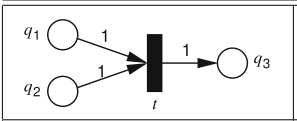
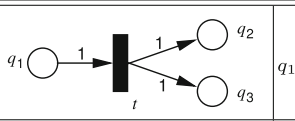
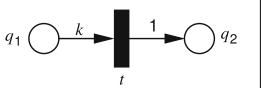
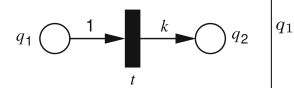
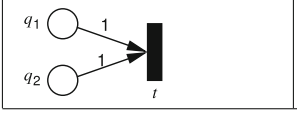
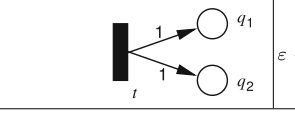
Fig. 2. Inference rules for the definition of the derivation relation in PRSs.

Remark 1. For this paper, we use always $F \triangleq \{\varepsilon\}$

Definition 2 (Normal Form). *A* process algebraic expression $\pi \in PEX(Q)$ *of a* Process Rewrite System Π *is a* normal form *iff there exists no* process algebraic expression $\pi' \in PEX(Q)$ *with* $\pi \Rightarrow \pi'$.

For this work, the left-hand and the right-hand side of a process rewrite rule contains an atomic, parallel or sequential process. So both sides are process algebraic expressions without any restriction. If the right- and left-hand side only allows parallel processes, then there is a one-to-one correspondence to Petri-Nets (Place/Transition Nets) [17]. Table 1 sketches this correspondence. A parallel operator on the left-hand side of a process-rewrite rule corresponds to a synchronization, a parallel operator on the right-hand side of a process-rewrite

Table 1. Correspondence between Petri-Nets and PRS (only parallel processes on LHS and RHS of Rewrite Rules.)

Synchronization		Fork	
Petri-Net	PRS-rule	Petri-Net	PRS-rule
	$q_1 \parallel q_2 \xrightarrow{t} q_3$		$q_1 \xrightarrow{t} q_2 \parallel q_3$
	$\underbrace{q_1 \parallel \cdots \parallel q_1}_{k \text{ times}} \xrightarrow{t} q_2$		$q_1 \xrightarrow{t} \underbrace{q_2 \parallel \cdots \parallel q_2}_{k \text{ times}}$
	$q_1 \parallel q_2 \xrightarrow{t} \varepsilon$		$\varepsilon \xrightarrow{t} q_1 \parallel q_2$

rule to a fork of parallel processes (e.g. asynchronous service call), a sequential operator on the right-hand side corresponds to a procedure call or synchronous (blocking) service call, and a procedure return corresponds to a process-rewrite rule $q \to \varepsilon$ [13]. This work shows a one-to-one correspondence between general process-algebraic expressions and cactus stacks, i.e., an abstract program semantics can be viewed as transitions on cactus stacks, see Sect. 3. This is a well-known semantics of programs with parallel processes and were first introduced by Dahl and Nygaard for the runtime system of *Simula* (as *tree of stacks*) [5].

2.3 Deadlock Detection

A deadlock for Petri-Nets corresponds to a normal form (different from ε), i.e., a process-algebraic expression where no process-rewrite rule is applicable:

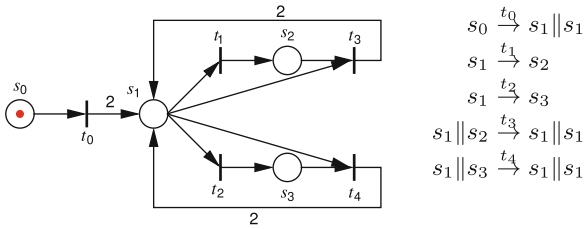


Fig. 3. A Petri-Net and its corresponding PRS.

Example 1. Figure 3 shows a Petri-Net that may lead to deadlock. This happens if the transitions fire in the order t_0, t_1, t_2 (leaving one token in place s_2 and one in place s_3). On the PRS, this corresponds to the derivation

$$s_0 \overset{t_0}{\Rightarrow} s_1 \| s_1 \overset{t_1}{\Rightarrow} s_2 \| s_1 \overset{t_2}{\Rightarrow} s_2 \| s_3$$

Now, no rule is applicable, i.e., the expression $s_2 \| s_3$ is a normal form. Note that there is an infinite firing sequence: $t_0, t_1, t_3, t_1, t_3, \ldots$. This corresponds to the derivation

$$s_0 \overset{t_0}{\Rightarrow} s_1 \| s_1 \overset{t_1}{\Rightarrow} s_2 \| s_1 \overset{t_3}{\Rightarrow} s_1 \| s_1 \overset{t_1}{\Rightarrow} s_2 \| s_1 \overset{t_3}{\Rightarrow} s_1 \| s_1 \cdots$$

According to the above discussions, this class of PRS is a generalization of Petri-Nets as it allows on the right-hand side the sequential operator which corresponds to procedure calls including (mutually) recursive procedures. Let $\Pi \triangleq (\Sigma, Q, \to, q_0, \{\varepsilon\})$ a process-rewrite system and $\pi \in PEX(Q)$ reachable from q_0, i.e., $q_0 \overset{*}{\Rightarrow} \pi$. π is a *deadlock* for Π iff $\pi \neq \varepsilon$ and π is in normal form. [18] shows that deadlock detection is decidable for the class of PRS. Hence, it is possible to use PRSs for deadlock detection on services if recursion without restriction on the recursion depth and parallelism without restricting the degree of parallelism is allowed.

2.4 Abstraction

This section shows how to abstract the behaviour of services to process rewrite systems.

The service-oriented system in Fig. 4 includes a synchronous and asynchronous (forking) procedure call, a synchronization and recursion. The used states correspond to the program points of the implementation of the services.

Table 2. Abstraction of control structures to process rewrite rules [12].

Control structure	Abstraction	Explanation
$q : n := 1; q'$	$q \rightarrow q'$	Assignments have no influence on deadlock behaviour. q' is the program point of the statement being executed after $n := 1$
$q : if(e) q' : \ldots$ $else\, q'' : \ldots$ q'''	$q \rightarrow q'$ $q \rightarrow q''$	No influence on deadlock behaviour if the condition is being decided. q''' is the program point of the statement being executed after the last statement of the then- and else-part
$q : p(\ldots); q' : \ldots$ $\vdots$ $p(\ldots)\{q'' : \ldots\}$	$q \rightarrow q''.q'$	Call of a synchronous procedure p: The program point q' of the statement to be executed after the call is pushed onto the stack. The execution continues with first program point q'' of p
$q : p(\ldots); q'.$ $\vdots$ $p(\ldots)\{q'' : \ldots\}$	$q \rightarrow q''\|\|q'$	Call of an asynchronous procedure p: The execution can be continued concurrently with the statement at program point q' after the call and the statement at the first program point q'' and q'
$p(\ldots)\{\ldots$ $q'' : \mathbf{return};\}$	$q''.q' \rightarrow q'$	The current synchronous called procedure is left and the execution continues with the statement after the call. The corresponding program point q' was pushed upon call, cf. [13]
$p(\ldots)\{\ldots$ $q'' : \mathbf{return};\}$	$q''\|\|q' \rightarrow q'$ $q'\|\|q'' \rightarrow q'$	The current asynchronous called procedure is left and the execution continues with the statement after the call q'. The forked execution is being joined
$q : sync\, p; q' : \ldots$ $\vdots$ $p(\ldots)\{\ldots$ $q'' : \mathbf{return};\}$	$q\|\|q'' \rightarrow q'$	The statement after q (at program point q') can only be executed when the previously called asynchronously procedure p returns

Table 2 shows how the different control structures of our example can be abstracted. By applying these rules to the service oriented system in Fig. 4 the shown process rewrite rules $\rightarrow$ in Table 3 can be abstracted. A derivation from q_0 (the initial state of the abstraction Π) is shown in Table 4.

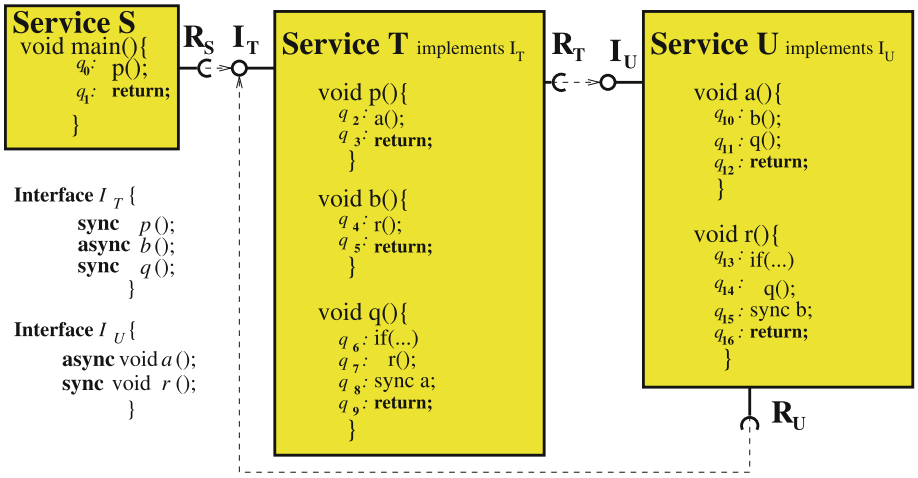


Fig. 4. A service-oriented system with three services. Service S acts as a client.

Table 3. Abstraction of the example of Fig. 4 to process rewrite rules $\rightarrow$.

Abstracted PRS-rules
(1) $q_0 \rightarrow q_2.q_1$ **(2)** $q_1 \rightarrow \epsilon$ **(3)** $q_2 \rightarrow q_3 \| q_{10}$ **(4)** $q_3 \| q_{12} \rightarrow q_3$ **(5)** $q_3.q_1 \rightarrow q_1$ **(6)** $q_4 \rightarrow q_{13}.q_5$ **(7)** $q_6 \rightarrow q_7$ **(8)** $q_6 \rightarrow q_8$ **(9)** $q_7 \rightarrow q_{13}.q_8$ **(10)** $q_{12} \| q_8 \rightarrow q_9$ **(11)** $q_9.q_{12} \rightarrow q_{12}$ **(12)** $q_9.q_{15} \rightarrow q_{15}$ **(13)** $q_{10} \rightarrow q_4 \| q_{11}$ **(14)** $q_{11} \rightarrow q_6.q_{12}$ **(15)** $q_5 \| q_{12} \rightarrow q_{12}$ **(16)** $q_{12} \| q_8 \rightarrow q_9$ **(17)** $q_{13} \rightarrow q_{14}$ **(18)** $q_{13} \rightarrow q_{15}$ **(19)** $q_{14} \rightarrow q_6.q_{15}$ **(20)** $q_5 \| q_{15} \rightarrow q_{16}$ **(21)** $q_{16}.q_5 \rightarrow q_5$ **(22)** $q_{16}.q_8 \rightarrow q_8$

Table 4. Derivation of example of Fig. 4 with the process rewrite rules $\rightarrow$ from Table 3.

Derivation of PRS-rules	Rule	Represents Method call
$q_0 \Rightarrow q_2.q_1$	(1)	$p()$;
$\underline{q_2}.q_1 \Rightarrow (q_3 \| q_{10}).q_1$	(3)	$a()$;
$(q_3 \| \underline{q_{10}}).q_1 \Rightarrow (q_3 \| q_4 \| q_{11}).q_1$	(13)	$b()$;
$(q_3 \| \underline{q_4} \| q_{11}).q_1 \Rightarrow (q_3 \| q_{13}.q_5 \| q_{11}).q_1$	(6)	$r()$;
$(q_3 \| \underline{q_{13}}.q_5 \| q_{11}).q_1 \Rightarrow (q_3 \| q_{14}.q_5 \| q_{11}).q_1$	(17)	$if(...)$
$(q_3 \| q_{14}.q_5 \| q_{11}).q_1 \Rightarrow (q_3 \| q_6.q_{15}.q_5 \| q_{11}).q_1$	(19)	$q()$;
$(q_3 \| \underline{q_6}.q_{15}.q_5 \| q_{11}).q_1 \Rightarrow (q_3 \| q_8.q_{15}.q_5 \| q_{11}).q_1$	(8)	$if(...)$
$(q_3 \| q_8.q_{15}.q_5 \| \underline{q_{11}}).q_1 \Rightarrow (q_3 \| q_8.q_{15}.q_5 \| q_6.q_{12}).q_1$	(14)	$q()$;
$(q_3 \| q_8.q_{15}.q_5 \| \underline{q_6}.q_{12}).q_1 \Rightarrow (q_3 \| q_8.q_{15}.q_5 \| q_8.q_{12}).q_1$	(8)	$if(...)$
$(q_3 \| q_8.q_{15}.q_5 \| q_8.q_{12}).q_1 \not\Rightarrow$	–	**normal form**

By applying the rules $1, 3, 13, 6, 17, 19, 8, 14, 8$ as shown in Table 4 we maintain the process algebraic expression $\pi = (q_3 \| q_8.q_{15}.q_5 \| q_8.q_{12}).q_1$. It represents a normal form because no rules can be applied. So, the service oriented system reaches a *deadlock* when entering state π.

Remark 2. The abstraction is similar to control-flow graphs as used for program analysis [2]. Each standard statement has a unique entry point but possibly multiple exit points (e.g. break and continue-statements). For simplicity, Table 2 only considers statements with a single exit point. Multiple Exit points are future work. Decisions in conditionals and loops could be abstracted to nondeterministic choices and belongs to future work. For asynchronous procedure calls, the place after the call is considered as the unique exit point.

Remark 3. The abstraction of every service to PRS Rules $\rightarrow$ is done by following the abstraction rules of Table. 2. Every service abstraction will be combined to a set of PRS rules of the system, shown in Table 3.

3 Capturing Recursion and Concurrency

An one-to-one correspondence between general process-algebraic expressions and cactus stacks, i.e., an abstract program semantics can be viewed as transitions on cactus stacks. This is a well-known semantics of programs with parallel processes and were first introduced by Dahl and Nygaard for the runtime system of *Simula* (as *tree of stacks*) [5].

Qur approach captures both, parallel and sequential behaviour. However, to show the one-to-one correspondence the corresponding cactus stack to the derivation of the Example can be seen in Table 5.

By applying rule (3) to the (cactus) stack in line 1, the top element q_0 is replaced by q_1 and q_2 (pop q_0 out of the stack, push q_1 followed by q_2 into the stack). This behaviour represents the sequential behaviour. At program point q_0 the synchronous procedure p of $Service\,T$ is called. So program point q_1, the program point after q_0, is pushed on the stack and on top of q_1 the first program point of the procedure p, q_2. The element q_1 can only be removed from the stack, when the synchronous method p returns, which means q_2 is removed from the stack.

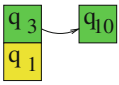
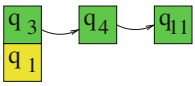
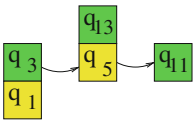
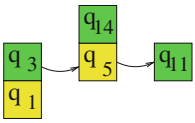
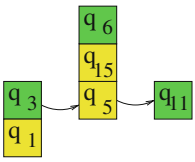
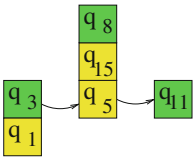
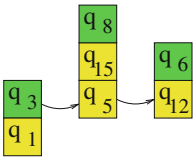
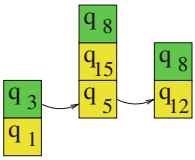
By applying rule (13) to the cactus stack in line 3, the top element q_{10} is removed (pop) and replaced by element q_4 (pop and push on an empty stack branch). However, another stack branch is created with q_{11} as top element. The new branch represents the parallel behaviour of Example Fig. 4. Here, the asynchronous method b of $Service\,T$ is called. The first program point of $b()$ is q_4. The first program point of $a()$ is q_{10}.

Remark 4. Rules can only be applied on top of the stack or stack branches. In Table 5 these are the green elements. Compared to the algebraic expression, these are the program points on the top left of the sequential part.

4 Related Work

In [22] recursive Petri Nets (rPN) are used to model the planning of autonomous agents which transport goods form location A to location B and their coordinating problem. The model of recursive Petri Nets is used to model dynamic

Table 5. The one-to-one correspondence between general process algebraic expressions and cactus stacks shown using the example of Fig. 4, cf. derivation shown in Table 4.

Algebraic Expression	Applicable Rule	Corresponding Cactus Stack
q_0	(1) $q_0 \rightarrow q_2.q_1$	
$q_2.q_1$	(3) $q_2 \rightarrow q_3 \vert\vert q_{10}$	
$(q_3 \vert\vert q_{10}).q_1$	(13) $q_{10} \rightarrow q_4 \vert\vert q_{11}$	
$(q_3 \vert\vert q_4 \vert\vert q_{11}).q_1$	(6) $q_4 \rightarrow q_{13}.q_5$	
$(q_3 \vert\vert q_{13}.q_5 \vert\vert q_{11}).q_1$	(17) $q_{13} \rightarrow q_{14}$	
$(q_3 \vert\vert q_{14}.q_5 \vert\vert q_{11}).q_1$	(19) $q_{14} \rightarrow q_6.q_{15}$	
$(q_3 \vert\vert q_6.q_{15}.q_5 \vert\vert q_{11}).q_1$	(8) $q_6 \rightarrow q_8$	
$(q_3 \vert\vert q_8.q_{15}.q_5 \vert\vert q_{11}).q_1$	(14) $q_{11} \rightarrow q_6.q_{12}$	
$(q_3 \vert\vert q_8.q_{15}.q_5 \vert\vert q_6.q_{12}).q_1$	(8) $q_6 \rightarrow q_8$	
$(q_3 \vert\vert q_8.q_{15}.q_5 \vert\vert q_8.q_{12}).q_1$	–	

processes (e.g., agent's request). Recursion in our sense is not considered. Deadlocks can only arise when interactions between agents (e.g., shared attributes) invalidates preconditions. For that reason a coordinating algorithm is introduced to prevent deadlocks (interactions between agents).

Another refinement based approach is described in [14]. This approach models healthcare processes and its called sub-processes with recursive Petri Nets.

Recursion in our sense is not considered. Both approaches use the ability of rPNs to prune sub trees from the state. In [10,14] recursion is used to allow the flexible extension of a certain workflow, e.g. health care workflow with a flexible healthcare process, called sub-processes.

[3] uses also Process Rewrite Systems to check the right behaviour of service-oriented Systems. Instead of considering deadlocks to ensure the right behaviour they check protocol conformance. Here, too, recursion and concurrency (internal and over service boundaries) of the services in a service-oriented Systems are allowed. Protocols ensure the right call sequence of all callable operations defined in an interface of a single service or instance. Like ours, this approach is abstraction based.

[1] uses workflow-nets to model business processes. This approach is based on Petri Nets. Hence only concurrency can be considered.

An abstraction based approach is done by Bouajjani et al. [4]. They discuss the analysis of recursive parallel programs based on recursive vector addition systems. They also explore the decidability of problems such as reachability. It seems that their model is slightly more general as there are situations where the reachability problem becomes undecidable. Neither deadlock analysis in services nor in composed systems is considered. Another abstraction based approaches are done by [6,7]. Both approaches are based on *lam programs*, a JAVA-like language introduced in [8]. [6] uses a points-to analysis and a may-happen-in-parallel-analysis to build up a dependency graph. The absence of cycles in the graph ensures deadlock freeness. The challenging example is how to handle object and task creation in loops.

The approach of [7] is based on a type system which is associated to the processes (or networks with nodes) of the lam program. The authors show that for a subset of lam programs including recursion of a non linear problem, deadlock-freeness is decidable. E.g., the recurrence relation of Fibonacci is a non linear and the recurrence relation of the factorial function is a linear problem.

Neither deadlock analysis in services nor in composed systems is considered. So deadlocks that appear by recursive callbacks can possibly not being detected. This is subject for further research.

To our knowledge, abstraction-based deadlock analysis in service-oriented systems including synchronous and asynchronous procedure calls (forking), recursion and recursive callbacks was not investigated before.

5 Research Plan

We want to examine in detail whether all possible processes within a service come to an end or all called processes in a composition of services, a service-oriented

system, come to an end. Services are allowed to communicate asynchronously and synchronously. Also, recursion and recursive callbacks are allowed. Furthermore, our goal was to state an approach that supports the black-box behaviour of services to keep the business secret.

However, there are two reasons why a service or a service-oriented system can not come to an end: the service or service-oriented system can not terminate (recursion/sequential case) or deadlocks (concurrency).

So we used termination and deadlock analysis to decide if the called processes of or a service-oriented system comes to an end or not.

In summary the following points were or will be considered:

1 determination of areas of improvement towards termination analysis of service-oriented systems [27].
2 termination analysis of service-oriented systems with no asynchronous calls and with no recursive callbacks [26,27].
3 termination analysis of service-oriented systems with recursive callbacks [28].
4 deadlock analysis of service-oriented systems with only concurrency including asynchronous calls [25], partly this paper.
5 termination and deadlock analysis of service-oriented systems including 3 and 4 (this paper).
6 implementation of an automatic verification tool and case study.

Literature research has shown that the common consideration of recursion and concurrency in termination and deadlock analysis in service-oriented systems is missing. In [27] we pointed out that deadlock analysis based on Petri Nets only abstract the concurrent behaviour. Recursion is ignored and fairness and soundness of the abstracted Petri Net is assumed. But under certain circumstances this can lead to a deadlock [27]. We proposed the introduction of a termination and size change function which can be provided over the WSDL-description of every service [27].

In [28] recurrence equations were introduced to handle recursion and recursive callbacks. With the help of a closed system of recurrences the termination can be proved.

There are still open points that need to be considered. For example under which circumstances, recursion can be ignored by doing the deadlock analysis.

6 Conclusion

An overview of the research topic deadlock analysis in service-oriented systems was given. We allow concurrency and recursion, internal and over service boundaries. Web Services can be described using Process Rewrite Systems. Process Rewrite Systems can model concurrency and recursion. The PRS description can be composed to describe the service behaviour and therefore enables the application of the deadlock detection algorithm for PRSs.

Recursive callbacks combined with concurrency, further implementation of a fully automatically tool to test the practical suitability and applicability are still

open points to be investigated. Since we have an abstraction based approach, it seems to be interesting if the verification can be done by only providing the abstractions of the service to keep the business secret and the black-box view.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63139-9_48
2. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Addison-Wesley Longman Publishing Co. Inc., Boston (2006)
3. Both, A., Zimmermann, W.: Automatic protocol conformance checking of recursive and parallel BPEL systems. In: IEEE Sixth European Conference on Web Services (ECOWS 2008), pp. 81–91 (2008)
4. Bouajjani, A., Echahed, R., Habermehl, P.: Verifying infinite state processes with sequential and parallel composition. In: Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 95–106. ACM (1995)
5. Dahl, O.J., Nygaard, K.: Simula: an algol-based simulation language. Commun. ACM **9**(9), 671–678 (1966)
6. Flores-Montoya, A.E., Albert, E., Genaim, S.: May-happen-in-parallel based deadlock analysis for concurrent objects. In: Beyer, D., Boreale, M. (eds.) FMOODS/-FORTE -2013. LNCS, vol. 7892, pp. 273–288. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38592-6_19
7. Giachino, E., Kobayashi, N., Laneve, C.: Deadlock analysis of unbounded process networks. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 63–77. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44584-6_6
8. Giachino, E., Laneve, C.: A beginner's guide to the Dead*L*ock *A*nalysis *M*odel. In: Palamidessi, C., Ryan, M.D. (eds.) TGC 2012. LNCS, vol. 8191, pp. 49–63. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41157-1_4
9. Haddad, S., Poitrenaud, D.: Theoretical aspects of recursive petri nets. In: Donatelli, S., Kleijn, J. (eds.) ICATPN 1999. LNCS, vol. 1639, pp. 228–247. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48745-X_14
10. Haddad, S., Poitrenaud, D.: Modelling and analyzing systems with recursive petri nets. In: Boel, R., Stremersch, G. (eds.) Discrete Event Systems, vol. 569, pp. 449–458. Springer, Boston (2000). https://doi.org/10.1007/978-1-4615-4493-7_48
11. Haddad, S., Poitrenaud, D.: Recursive petri nets. Acta Informatica **44**(7), 463–508 (2007). https://doi.org/10.1007/s00236-007-0055-y
12. Heike, C., Zimmermann, W., Both, A.: Protocol conformance checking of services with exceptions. In: De Paoli, F., Pimentel, E., Zavattaro, G. (eds.) ESOCC 2012. LNCS, vol. 7592, pp. 122–137. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33427-6_9
13. Heike, C., Zimmermann, W., Both, A.: On expanding protocol conformance checking to exception handling. Serv. Oriented Comput. Appl. **8**(4), 299–322 (2014)
14. Hicheur, A., Ben Dhieb, A., Barkaoui, K.: Modelling and analysis of flexible healthcare processes based on algebraic and recursive petri nets. In: Weber, J., Perseil, I. (eds.) FHIES 2012. LNCS, vol. 7789, pp. 1–18. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39088-3_1

15. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation, 2nd edition. SIGACT News **32**(1), 60–65 (2001). https://doi.org/10.1145/568438.568455
16. Kaveh, N., Emmerich, W.: Deadlock detection in distribution object systems. In: ACM SIGSOFT Software Engineering Notes, vol. 26, pp. 44–51. ACM (2001)
17. Mayr, R.: Combining petri nets and PA-processes. In: Abadi, M., Ito, T. (eds.) TACS 1997. LNCS, vol. 1281, pp. 547–561. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0014567
18. Mayr, R.: Decidability and complexity of model checking problems for infinite-state systems. Citeseer (1998)
19. Parizek, P., Plasil, F.: Modeling of component environment in presence of callbacks and autonomous activities. In: Paige, R.F., Meyer, B. (eds.) TOOLS EUROPE 2008. LNBIP, vol. 11, pp. 2–21. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69824-1_2
20. Rai, G.N., Gangadharan, G., Padmanabhan, V.: Algebraic modeling and verification of web service composition. Procedia Comput. Sci. **52**, 675–679 (2015)
21. Schmidt, H.W., Krämer, B.J., Poernomo, I., Reussner, R.: Predictable component architectures using dependent finite state machines. In: Wirsing, M., Knapp, A., Balsamo, S. (eds.) RISSEF 2002. LNCS, vol. 2941, pp. 310–324. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24626-8_22
22. Seghrouchni, A.E.F., Haddad, S.: A recursive model for distributed planning. In: Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS 1996), pp. 307–314 (1996)
23. van der Aalst, W.M.P.: Workflow verification: finding control-flow errors using petri-net-based techniques. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45594-9_11
24. Verbeek, E., van der Aalst, W.M.P.: Woflan 2.0 a petri-net-based workflow diagnosis tool. In: Nielsen, M., Simpson, D. (eds.) ICATPN 2000. LNCS, vol. 1825, pp. 475–484. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44988-4_28
25. Weißbach, M.: Termination analysis of concurrent service-oriented systems. Technical report, Proceedings of the Ph.D. Symposium at the 4th European Conference on Service-Oriented and Cloud Computing 1(01), pp. 23–29 (2015)
26. Weißbach, M., Zimmermann, W.: Checking of liveness properties in component-based systems and sercvice-oriented architectures. Technical report, Proceedings of the Ph.D. Symposium at the 8th IEEE European Conference on Web Services 1(4), pp. 09–12 (2010)
27. Weißbach, M., Zimmermann, W.: Termination analysis of business process workflows. In: Proceedings of the 5th International Workshop on Enhanced Web Service Technologies, WEWST 2010, pp. 18–25. ACM, New York (2010). https://doi.org/10.1145/1883133.1883137
28. Weißbach, M., Zimmermann, W.: Termination analysis of service-oriented systems. Technical report, Proceedings of the Ph.D. Symposium at the 1st European Conference on Service-Oriented and Cloud Computing 1(3), pp. 23–38 (2012)

Prediction of Quality of Service of Software Applications

Ahmad Ibrahim(✉)

Department of Computer Science,
University of Pisa, Pisa, Italy
ahmad@di.unipi.it

Abstract. The ability to a priori predict the Quality of Service (QoS) of a software application is crucial both in the design of applications and in the definition of their Service Level Agreements (SLA). QoS prediction is challenging because of the different possible results of service invocations, and of the nondeterminism, correlations and complex dependencies among activities.

In this research we present a technique to probabilistically predict the QoS of service based and parallel design patterns based applications by applying Monte Carlo simulations to a simple representation of the control-flow of the applications. A proof-of-concept implementation of the analyses is discussed along with future work.

1 Introduction

Quality of Service (QoS) refers to a set of non-functional attributes used to describe a system [1] such as response time, reliability or cost.

The QoS of an application does depend on the QoS of the services it uses. For instance, for service based applications [2,3], the response time of a service orchestration obviously depends on the response times of the services it invokes - which may actually vary over time for different reasons (e.g., workload or network congestion).

A straightforward way to assess the QoS of an application would be to simply deploy it and then monitor the QoS parameters of interest over a sufficiently high number of runs. Unfortunately, such an approach can be time consuming, expensive (when non-free services are invoked) and may not be feasible when invocations have side-effects (e.g., as in the case of services enabling monetary transactions).

In this research we consider two classes of applications

1. *Service based applications.* Service based application are usually defined via workflows that implement business processes by orchestrating various (possibly third-party) services [2,3].

Supervisor: Antonio Brogi, Department of Computer Science, University of Pisa, Italy.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 260–273, 2018.
https://doi.org/10.1007/978-3-319-72125-5_21

2. *Parallel design patterns based applications.* Parallel design patterns (also called skeletons) are customisable, composable design patterns that can be fruitfully exploited to define parallel applications [4,5].

The ability of a priori predicting the QoS of an application is very valuable for application designers. In service based applications, predicting the QoS of a service orchestration is valuable both during the design of the orchestration and for defining its Service Level Agreement (SLA [6]). It helps answering questions, like:

- What is the QoS of a given orchestration?
- What is the effect on the QoS of an orchestration of replacing one or more of the invoked services with alternative services, (e.g., offered by different providers)?
- How modifying the workflow of an orchestration impacts on its overall QoS?

Similarly, for parallel design pattern based applications, QoS prediction is valuable for comparing different deployments as well as for assessing the scalability of their parallelization. It helps answering questions, like:

- What is the QoS of a given parallel design pattern based application?
- Would a restructuring improve the QoS of a parallel design pattern based application for a steady state?
- What type of parallelization would achieve the best QoS for a dynamic state?

Challenges in predicting the QoS of an application

A priori predicting the QoS of an application is not easy mainly because of four main challenges.

1. ***Different possible results of service invocations.*** In *service based applications*, each invoked service can return a successful reply, a fault notification, or even no reply at all [7]. If a fault is returned, a fault handling routine will be executed instead of the normal control flow. If no reply is received, the orchestrator will get stuck (unless some parallel branch throws a fault). In either case, the resulting QoS of the orchestration will differ from the case of successful invocation.
   In the case of parallel design patterns based applications, the QoS of the activities executed depends for instance on the type of the inputs arriving in the input stream, as different inputs typically require different service times.
2. ***Non-determinism in the orchestration workflow.*** The control flow of a workflow defining a *service based applications* is in general nondeterministic. Besides the nondeterminism induced by the possible different results of service invocations, a workflow usually contains branching conditions that depend upon input data values. As a consequence, which branch will be executed in an alternative, or the number of iterations of a loop is not known a priori [8,9].

In the case of parallel design patterns based applications, the execution of conditional loops defined by `feedback` patterns has a similar nondeterministic behaviour, as some outputs may be routed back depending upon the evaluation of conditions, and the number of iterations is not known a priori.

3. ***Complex dependencies among workflow activities.*** Workflows can contain complex dependencies among activities, as for instance those defined by WS-BPEL synchronization `links` [10]. The control flow defined by synchronizations *within* parallel activities is more expressive than what is allowed by simple parallel execution with synchronization barriers at the end. This implies that workflows containing such complex synchronization structures cannot be always decomposed into parallel and sequential compositions, as shown in [11,12].
   In the case of parallel design patterns based applications, the execution of a `pipe` pattern for instance exhibits a complex dependencies behaviour. An activity in a `pipe` can proceed with processing new input and run in parallel along with other activities which can be still busy with previous inputs.
4. ***Correlations among workflow activities.*** The above two characteristics suggest to employ a probabilistic approach. However, it is important to observe that the naive solution of assigning independent probabilities to workflow activities (e.g., as in [11]) can lead to incorrect results.

To the best of our knowledge, none of the existing techniques for QoS prediction for service based (e.g., [11–16]) and parallel design pattern based applications (e.g., [17–20]) fully addresses all the aforementioned challenges respectively.

2 Research Contributions

In this research we present a design time QoS prediction algorithm for service based and parallel design patterns based applications. The QoS prediction techniques rely on two main ideas:

(1) Expressing the control flow of applications in terms of two simple cost compositors (`Both` and `Delay`) to address complex dependency, and
(2) Exploiting Monte Carlo simulations [21] to deal with the non-determinism, different possible results of service invocations and correlations in workflow.

For both *service based applications* and *parallel design patterns based applications*, we present algorithms that suitably deals with all the challenges and that is capable of probabilistically predicting the QoS of an application.

3 Proposed Solution

To estimate the QoS of parallel design patterns based applications, we proposed structurally recursive functions that associate, in a compositional way, each activity with a cost structure. Such cost structure is general, and it can be instantiated to define different QoS attributes, e.g., the time needed to complete an activity etc. The cost compositors are:

– The first cost compositor is a parallel compositor. `Both(A,B)` defines the cost associated with executing independently an activity with cost A and an activity with cost B.
– The second cost compositor defines the delayed cost of an activity which must wait for the completion of another activity before starting. `Delay(A,B)` defines the cost associated with executing an activity of cost A which must wait for the completion of an activity with cost B before starting.

For example, let us denote with `aTime` and `bTime` the completion time of the two activities. The completion time for running both activities in parallel is given by the maximum between `aTime` and `bTime`. Instead, the completion time for delaying one activity after the other is obtained by summing `aTime` and `bTime`, as the delayed activity can start only after the first activity is completed.

```
let Both(aTime,bTime) = Max(aTime,bTime)
let Delay(aTime,bTime) = aTime + bTime
```

Other QoS properties (like energy consumption, monetary cost) can be similarly defined using `Both` and `Delay`. Our algorithms respectively converts a given workflow into an expression of `Both` and `Delay` and then estimate the QoS value.

3.1 Service Based Applications

For *service based applications*, the algorithm converts a given WS-BPEL workflow into an expression of `Both` and `Delay`.

The algorithm [22–24] is implemented in a open source tool PASO (Probabilistic Analyser of Service Orchestrations)[1]. The inputs of the PASO (Fig. 1) are a WS-BPEL workflow[2], and probability distributions for the QoS properties of the invoked services as well as for the evaluation of the workflow branching conditions. The output of the algorithm is a probability distribution for the QoS properties (reliability, time and cost) of the orchestration.

A summary of the algorithm is given in Table 1.

3.2 Parallel Design Patterns Based Applications

For *parallel design patterns based applications*, the algorithm converts a given parallel design pattern based application into an expression of `Both` and `Delay`. The algorithm [25] is implemented in a open source tool PASA (Probabilistic Analyser of Skeleton-based Applications)[3] (Fig. 2).

PASA inputs (i) the description of a parallel application defined as a combination of basic activities (i.e., `Nodes`) and of the core stream parallel design patterns `Comp`, `Pipe`, `Farm`, `Feedback`, (ii) the size and the optional classification

[1] The source code of PASO is available at https://github.com/upi-bpel/paso.
[2] PASO is able to analyse a subset of WS-BPEL structural (`sequence`, `flow`, `if`, `while`, `scope`, and `faultHandlers`) and basic (`invoke`, `assign`, `receive`, `reply`) activities.
[3] The source code of PASA is available at https://github.com/ahmad1245/PASA.

Table 1. Overview of prediction algorithm for *service based applications*.

Workflow activity	Cost compositor syntax	Description
`Sequence(A,B)`	`Both(A, Delay(B,A))`	B will executed after A
`Flow(activitySet,linkSet)`	`Both(activitySet)`	All activities in activitySet will be executed in Parallel depending upon the links 1. Sort activities using links to ensure that preceding activities are evaluated before current activity 2. For all activities in Flow: 2.1 Identify all preceding activities of the current activity 2.2 Compute outcome of all preceding activities (Fault has precedence over stuck and success. Stuck has precedence over success) 2.3 Compute cost of all preceding activities 2.4 Evaluate join condition for current activity 2.5 Evaluate transition condition for all outgoing links of current activity 3 Compute cost and outcome of Flow once all activities have been analyzed
`Scope(A,faultHandler)`	`Both(A,Delay(faultHandler,A))` or `Both(A,Delay(Zero,A))`	If evaluation of A yields a Fault, then faultHandler is evaluated Otherwise cost of executing A is returned only
`Assign(name,expr)`	`Zero`	We assume that Assign has Zero cost
`IfThenElse(guard,A,B)`	`Both(A, Delay(zero,A))` or `Both(B, Delay(zero,B))`	Evaluates guard condition using sampling function If true, evaluate A else evaluate B
`While(guard,A)`	`Sequence (A,While(guard,A))` or `Zero`	Evaluates guard condition using sampling function If false, the body of the loop is skipped Otherwise evaluate the body of the loop and While again
`Invoke(partnerLink)`	–	Use sampling function to sample a value for QoS properties and Success/Fault/Stuck outcome

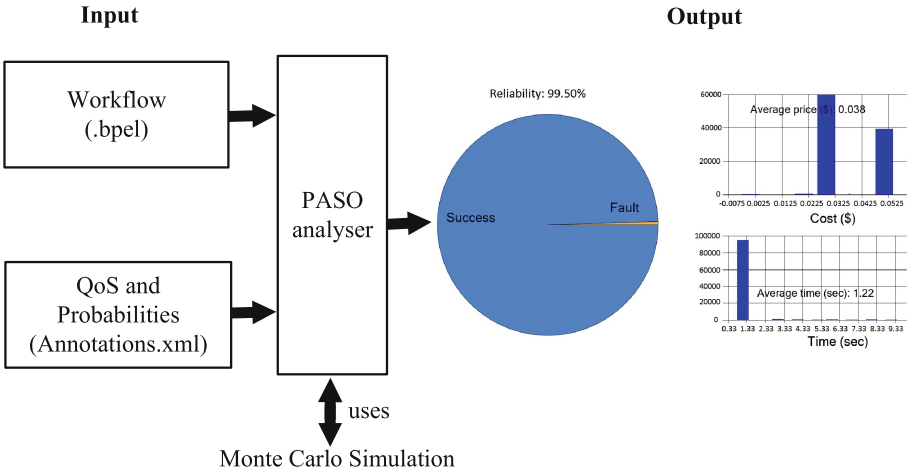


Fig. 1. Bird-eye view of the input-output behaviour of PASO.

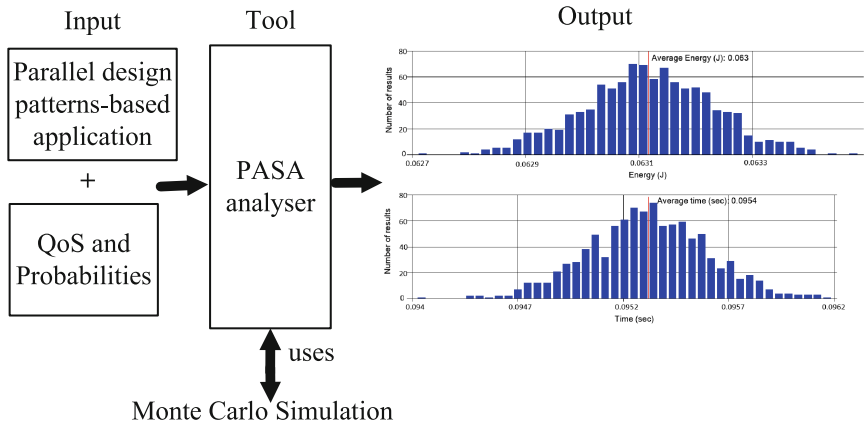


Fig. 2. Bird-eye view of the input-output behaviour of the PASA analyser.

of the input stream (e.g., we can categorise the data items coming from the input stream to distinguish the energy consumption and completion time they require for being processed), (iii) the QoS required by each node to process each type of input, and (iv) the probabilities of a given input type to occur and of a given `Feedback` conditions to get satisfied. PASA also permits displaying the results (i.e., energy consumption and completion time) of the performed analysis in different formats (e.g., histograms).

A summary of the algorithm is given in Table 2. The algorithms for both PASO and PASA are implemented in F#.Net [26].

Table 2. Overview of prediction algorithm for *parallel design patterns based applications.*

Pattern	Cost compositor syntax	Description
`Pipe (A, B)`	`Both (A, B)`	A and B will be executed simultaneously. More precisely, B will wait for A to process an input i before executing it. While B will be busy processing i, A could proceed with input i+1
`Comp (A, B)`	`Both (A, Delay(B,A))`	B can start processing a given input only when A has completed processing such input
`Farm (A, n)`	`Both (wQoS of n workers)`	We split the input stream among n workers Execute A on n workers simultaneously and save cost in wQoS
`Feedback(A, Condition)`	`Both(A,Delay(fQoS,A))`	Execute A & save cost in feedbackQoS Evaluate Condition using Sampling function If True then item is routed back, A is executed again & cost is saved in fQoS

4 Example

To illustrate our approach, we describe next its application to shipping service example, adapted from well-known example provided in the original WS-BPEL specification [10]. Although this example only shows the application of PASO, yet other examples for both PASO and PASA can be found in [23,25].

4.1 Shipping Service

The shipping service (Fig. 3) starts by receiving a shipping order from the customer. A shipping order contains a list of requested items together with the indication of whether the requested items must be shipped separately or all together. The service ships the orders as indicated and then it replies to the customer with a shipment complete notification.

We assume the following input distributions:

- Workflow control-flow (Table 3(a)): Condition *ShipIndividual* true 70% of times, condition $Item<TotalItems$ true 80% of times.
- ShipItem service (Table 3(b)): It usually (80%) completes in 2 s with 0.5$ expense. Sometimes (20%) it returns a fault after 3 s.

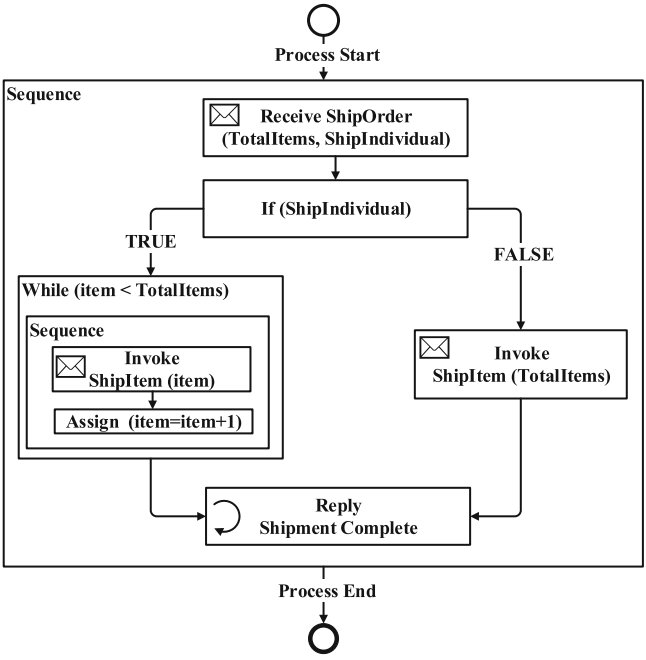


Fig. 3. Shipping service.

4.2 PASO at Work: Shipping Service

We now describe step-by-step analysis of PASO for the shipping service example. For instance, if PASO samples the value `false` for `ShipIndividual` and `<Success,0.5$,2 sec>` for the invocation, then the total *cost* of the workflow is:

```
Both(Zero,Both(<0.5$,2sec>,Both(Zero,Zero)))
    = <0.5$,2sec>
```

On the other hand, if PASO samples the value `false` for `ShipIndividual` and `<Fault,0$,3 sec>` for the invocation, then the total cost of the workflow is:

```
Both(Zero,Both(<0$,3sec>,Both(Zero,Zero)))
```

Table 3. Input distributions for the shipping service example.

condition	*p*
ShipIndividual	70%
item<TotalItems	80%

(a) Wokflow control-flow

outcome	*expense*	*time*	*p*
Success	0.5$	2 sec	80%
Fault	0.0$	3 sec	20%

(b) ShipItem service

```
= <0$,3sec>
```

If instead PASO samples the value `true` for `ShipIndividual` and the body of the `While` is executed three times with `<Success,0.5$,2 sec>` for the invocation, then the *cost* of the `While`, which is also the *cost* of the whole workflow, is

```
Both(x,Delay(Both(x,Delay(x,x)),x))
    = <1.5$,6sec>
```

(where `x=<0.5$,2sec>`).

Table 4 summarizes the previously described three traces along with other three runs of the `exec` function on the example.

Table 4. Six runs of the shipping service example.

ShipIndividual	item<TotalItems	ShipItem	Orchestration
false		<Success,0.5$,2 sec>	<Success,0.5$,2 sec>
false		<Fault,0$,3 sec>	<Fault,0$,3 sec>
true	true	<Success,0.5$,2 sec>	
	true	<Success,0.5$,2 sec>	
	true	<Success,0.5$,2 sec>	
	false		<Success,1.5$,6 sec>
false		<Success,0.5$,2 sec>	<Success,0.5$,2 sec>
true	false		<Success,0$,0 sec>
true	true	<Fault,0$,3 sec>	<Fault,0$,3 sec>

The values of the QoS properties *reliability*, *amortized expense for successful execution* and *average response time* computed from the samples of Table 4 are:

$$\texttt{successTime} = (2+3+6+2+0+3)\ \texttt{sec} = 16\ \texttt{sec}$$
$$\texttt{totalExpense} = (0.5+0+1.5+0.5+0+0)\$ = 2.5\$$$
$$\texttt{reliability} = 4/6 = 66.6\%$$
$$\texttt{amortizedExpense} = (2.5\$)/4 = 0.6\$$$
$$\texttt{averageResponseTime} = (16\ \texttt{sec})/4 = 4\ \texttt{sec}$$

Table 5 shows how the estimations of the QoS parameters considered progressively converges by increasing the number of samples.

4.3 Experimental Results

To validate the results predicted by PASO, we implemented and deployed shipping service by using Apache ODE [27] and Tomcat server [28] on a local server. A Java client program was used to invoke and monitor *response time* of the orchestration (Fig. 4).

Table 5. QoS estimations for different number of samples for the shipping service example.

Samples	`reliability`	`amortizedExpense`	`averageResponseTime`
6	66.6%	0.6 $	4 s
100	68%	0.8 $	4.2 s
10,000	62.8%	0.75 $	4.1 s
1,000,000	62.9%	0.74 $	4.1 s

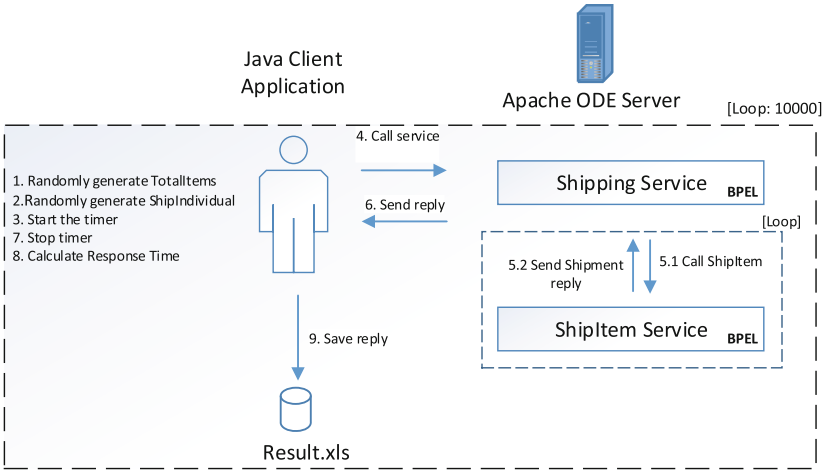


Fig. 4. Activity diagram for monitoring the shipping service example.

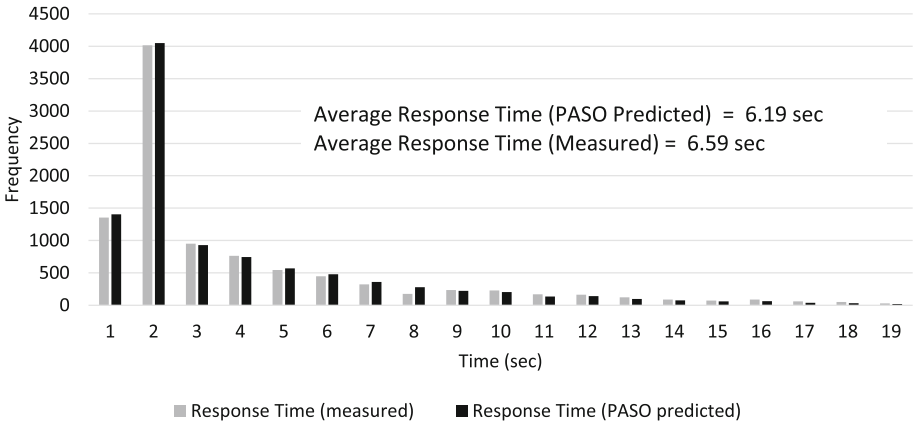


Fig. 5. Response time comparison for shipping service example

The results after monitoring *response time* for the shipping service are plotted, along with PASO predicted result, in Fig. 5 respectively. It is easy to see that PASO predicted results correspond to the monitored results. The slight variation in *response time* is due to the overhead associated with monitoring (which was not considered by PASO).

5 Related Work

Various approaches have been proposed to determine the QoS for service based applications (e.g., [11–16]) and parallel design patterns based applications (e.g., [17–20]).

5.1 Service Based Applications

Cardoso [13] presented a mathematical model and an algorithm to compute the QoS of a workflow composition. The stochastic workflow reduction algorithm iteratively reduces the workflow by removing parallel, sequence, alternative and looping structures according to a set of reduction rules, until only one activity remains. However, some workflow complex dependency structures cannot be decomposed into parallel or sequence, as shown in [29].

Mukherjee et al. [11] presented a algorithm to estimate the QoS of WS-BPEL compositions. They convert a WS-BPEL workflow into an activity dependency graph, and assign probabilities of being executed to each activity. Their framework allows complex dependency structure as well as *fault* driven flow control. However, [11] do not consider correlations among activities which do not have a direct dependency, and this in some cases can yield a wrong result.

Zheng et al. [15] focused on QoS estimation for compositions represented by service graphs. They transform the service graph in order to remove the cycles, then calculate probabilities of execution and QoS parameters for each path. In their approach however they only marginally deal with parallelism, by not considering arbitrary synchronization `<link>s` (i.e., they restrict to cases in which is possible to decompose *flow*-like structures into parallel and sequences, as in [13]), and they do not take into account fault handling.

Ivanovic et al. [30] defined a language to represent service compositions, and they address the problem of correlation. However the language does not describe parallel execution, thus their solution is similar to the ones proposed in workflow decomposition approaches [13,15].

Moreover, to the best of our knowledge, all previous approaches for QoS prediction for service based application require to know a priori the exact number of iterations, or at least an upper bound for each loop in order to estimate QoS values.

5.2 Parallel Design Patterns Based Applications

Benoit et al. [19] proposed using process algebras to predict QoS of skeleton applications. The main motivation was to analyze the performance of Grid applications with the use of algorithmic skeletons and process algebras. Their algorithm AMoGeT works by first generating PEPA models [31] from the given program. Solving the models and then comparing results provide performance information (*throughput*). Their approach only covered the *Pipe* pattern, though.

Castro et al. [20] proposed a QoS prediction algorithm based on denotational semantics. The skeletons they considered were *sequence*, *pipe* and *farm*. Their algorithm works in three steps. It starts by generating all possible alternatives for a given program. Then minimum number of worker threads for each alternatives is calculated. The alternative with lowest number of cores is selected at the end. The approach by [20] is deterministic and they assume that all activities will take same amount of time to execute, in their proposed cost model, which can be unrealistic in some scenario.

The approaches by Jay [17] and Hayashi et al. [18] covers only data parallel patterns and have limitations in terms of non-determinism like Castro et al. [20].

To summarize, only few approaches (e.g., [17–20]) focus on design time prediction of QoS for parallel design pattern based application. Most of them either focus only on data parallel patterns or limited number of stream parallel patterns (with no focus on non-deterministic scenarios).

6 Future Work

There are several possible directions to extend this work:

PASO and PASA can be extended to support new activities and languages. PASO is currently capable of analyzing a proper subset WS-BPEL. Support for other interesting WS-BPEL activities (e.g., `Pick`, `Event Handlers`) and workflow languages (e.g.,YAWL, BPMN) could be included in PASO. Similarly, PASA currently models a simple set of parallel design patterns. Support for other interesting data parallel patterns (e.g., `map`, `reduce`) and frameworks (e.g., FastFlow) could be included in PASA.

Some forms of correlations could be introduced in the samplings. For instance, it would be interesting to consider some correlation among service invocations (e.g., if a service invocation returns a fault because it is "down for maintenance" it may be probable that the same result will be obtained in the next invocation) and in the input stream of parallel applications (e.g., bursts of data).

Last, but not least, the definition of the cost compositors `Both` and `Delay` can be extended to support new QoS properties (e.g., throughput, availability, and so on). An interesting extension in this perspective would be to provide users with a query language to specify their own QoS properties.

Acknowledgments. This work was partly supported by the project *Through the Fog* (PRA_2016_64) funded by the University of Pisa.

References

1. ISO: CD 15935 - Information Technology: Open Distributed Processing - Reference Model - Quality of Service. Technical Report ISO document ISO/IEC JTC1/SC7 N1996, International Organization for Standardization (1998)
2. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R., Hamilton, B.A.: Reference model for service oriented architecture 1.0. OASIS standard **12** (2006)
3. Papazoglou, M.: Web Services: Principles and Technology, 2nd edn. Pearson Education Canada (2012)
4. Cole, M.I.: Algorithmic Skeletons: A Structured Approach to the Management of Parallel Computation. Ph.D. thesis, University of Edinburgh (1988)
5. Rabhi, F.A., Gorlatch, S. (eds.): Patterns and Skeletons for Parallel and Distributed Computing. Springer, London (2003)
6. Keller, A., Ludwig, H.: The WSLA framework: specifying and monitoring service level agreements for web services. J. Netw. Syst. Manage. **11**, 57–81 (2003)
7. Brüning, S., Weissleder, S., Malek, M.: A fault taxonomy for service-oriented architecture. In: 10th IEEE High Assurance Systems Engineering Symposium, HASE 2007, pp. 367–368 (2007)
8. Floyd, R.W.: Nondeterministic algorithms. J. ACM (JACM) **14**, 636–644 (1967)
9. Lohmann, N., Verbeek, E., Ouyang, C., Stahl, C.: Comparing and evaluating petri net semantics for BPEL. Int. J. Bus. Process Integr. Manag. **4**, 60–73 (2009)
10. Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al.: Web Services Business Process Execution Language version 2.0 (2007). http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
11. Mukherjee, D., Jalote, P., Gowri Nanda, M.: Determining QoS of WS-BPEL compositions. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 378–393. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89652-4_29
12. Dumas, M., García-Bañuelos, L., Polyvyanyy, A., Yang, Y., Zhang, L.: Aggregate quality of service computation for composite services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 213–227. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17358-5_15
13. Cardoso, A.J.S.: Quality of service and semantic composition of workflows. Ph.D. thesis, Univ. of Georgia (2002)
14. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. J. Syst. Softw. **82**, 3–22 (2009)
15. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: QoS analysis for Web service compositions with complex structures. IEEE Trans. Serv. Comput. **6**, 373–386 (2013)
16. Ivanović, D., Carro, M., Kaowichakorn, P.: Towards QoS prediction based on composition structure analysis and probabilistic models. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 394–402. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45391-9_29
17. Jay, C.B.: Costing parallel programs as a function of shapes. Sci. Comput. Program. **37**, 207–224 (2000)
18. Hayashi, Y., Cole, M.: Static performance prediction of skeletal parallel programs. Parallel Algorithms Appl. **17**, 59–84 (2002)

19. Benoit, A., Cole, M., Gilmore, S., Hillston, J.: Evaluating the performance of skeleton-based high level parallel programs. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 289–296. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24688-6_40
20. Castro, D., Hammond, K., Brady, E., Sarkar, S.: Structure, semantics and speedup: reasoning about structured parallel programs using dependent types. Under Consideration for Publication in J. Funct. Program. (2015)
21. Dunn, W.L., Shultis, J.K.: Exploring Monte Carlo Methods. Elsevier, Amsterdam (2011)
22. Bartoloni, L., Brogi, A., Ibrahim, A.: Probabilistic prediction of the QoS of service orchestrations: a truly compositional approach. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 378–385. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45391-9_27
23. Bartoloni, L., Brogi, A., Ibrahim, A.: Automated prediction of the QoS of service orchestrations: PASO at work. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 111–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33313-7_8
24. Bartoloni, L., Brogi, A., Ibrahim, A.: Predicting the QoS of service orchestrations. Submitted for publication, February 2016
25. Brogi, A., Danelutto, M., De Sensi, D., Ibrahim, A., Soldani, J., Torquati, M.: Analysing multiple QoS attributes in parallel design patterns-based applications. In: 9th International Symposium on High-Level Parallel Programming and Applications (HLPP 2016), Münster, Germany (2016)
26. Syme, D., Granicz, A., Cisternino, A.: Expert F# 3.0, 3rd edn. Apress, Berkeley (2012)
27. Apache Software Foundation: Apache ODE (Orchestration Director Engine) 1.3.6 (2013). http://ode.apache.org
28. Apache Software Foundation: Apache Tomcat 7.0.61 (2011). http://tomcat.apache.org
29. Mukherjee, D.: QoS in WS-BPEL Processes. Master's thesis, Indian Institute of Technology, Delhi (2008)
30. Ivanovic, D., Kaowichakorn, P., Carro, M.: Towards QoS prediction based on composition structure analysis and probabilistic environment models. In: 2013 ICSE Workshop on Principles of Engineering Service-Oriented Systems (PESOS), pp. 11–20. IEEE (2013)
31. Hillston, J.: A Compositional Approach to Performance Modelling, vol. 12. Cambridge University Press, Cambridge (2005)

Impact-Minimizing Runtime Adaptation in Cloud-Based Data Stream Processing

Cui Qin(✉)

Software Systems Engineering, University of Hildesheim,
Universitätsplatz 1, 31141 Hildesheim, Germany
qin@sse.uni-hildesheim.de

Abstract. Recently, cloud-based data stream processing has emerged to process huge amounts of data. During such processing, the actual characteristics of data streams may vary, e.g., in terms of volume or velocity. For example, in the financial domain hectic markets can cause bursty streams of events leading to changes of the stream characteristics by several orders of magnitude. To handle such situations, adaptation of the data processing at runtime is desirable. While several techniques for changing data stream processing at runtime do exist, one specific challenge is to minimize the impact of runtime adaptation on the data processing, in particular for real-time data analytics.

In this research work, we aim at performing runtime adaptation in cloud-based data stream processing, namely, dynamically switching alternative distributed algorithms, which have similar functionality, but operate at different characteristics (tradeoffs). The goal of this work is to provide a generic approach which can automatically determine the algorithm switch with minimized impact on the data processing. To achieve this goal, we introduce the concept of a "safe" (transparent, gap-free) switch, which takes the characteristics of alternative algorithms into account. For the actual switch, we combine stream re-routing with buffering and stream synchronization along with a support of dynamic deployment of alternative stream processing algorithms into the cloud.

Keywords: Cloud-based data stream processing
Runtime adaptation · Impact-minimizing adaptation
Algorithm switching · Dynamic deployment

1 Introduction

Big data applications aim at processing huge or complex data sets, which usually cannot be handled by traditional approaches. Distributed stream processing [3], i.e., continuous processing of conceptually endless streams of data items, is a popular approach to realize such applications. Particularly, cloud-based data stream processing, driven by the trend towards cloud computing, has rapidly

Supervisors: Klaus Schmid, Holger Eichelberger.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 274–285, 2018.
https://doi.org/10.1007/978-3-319-72125-5_22

emerged and become competitive in high scalability and robustness in terms of fault tolerance for processing huge amounts of data [15]. Well-known systems supporting cloud-based data stream processing include Apache Storm [2] and Apache Spark [1]. Such stream processing systems aim at real-time processing of complex and distributed stream analysis tasks.

During processing, the stream characteristics such as volume or volatility can vary over time. For example, in the financial domain hectic markets can cause bursty streams leading to changes of the stream characteristics by several orders of magnitude. In such situations, the current stream processing implementation may not withstand the occurring bursty streams due to the heavy workload. For instance, it can lead to a sudden slowdown of the processing, or even worse, cause data buffer-overflow and loss of data, in particular, with the utilization of buffering techniques in data stream processing. To cope with such problems, adaptation of the data processing at runtime is desirable.

A typical way to handle bursty streams is to allocate additional computing resources for the current stream processing. The *elasticity* of resource provisioning offered by the cloud computing model makes this possible, in particular, adapting the resource allocation at runtime. Representative approaches are proposed in [23] adapting the CPU allocation for data streaming applications under varying data arrival characteristics and [21] dynamically scaling up and down the processing to adjust the computational capacities to handle peaks and off-peaks. Such elastic data stream processing can directly benefit from the availability of public cloud infrastructure, which enables possibilities to provide virtually unlimited resources. However, current cloud service providers enable resource provisioning in a pay-as-you-go fashion, meaning that the utilization of computing resources from public cloud is directly visible to the monetary costs [14]. Also, with rather strict timing requirements in real-time data stream processing to dynamically allocate computing resources at runtime is still a chanllenging problem.

Other approaches such as Borealis [5] or RTSTREAM [25] provide adaptation capabilities for continuous stream processing queries, such as changing the query program. These approaches support the optimal execution of stream queries to deal with varying workloads, however, they concentrate on a fixed set of database-like stream operators and also do not support adaptation in a cloud-based environment. In contrast, recent frameworks such as Apache Storm or Spark provide possibilities for developers to implement complex analysis algorithms, e.g., realize distributed analysis tasks such as financial correlation computations in a scalable manner, but they currently do not provide much support for runtime adaptation.

In this research work, we support adapting stream processing in cloud-based environments by switching among different processing algorithms, which provide similar functionality but operate at different runtime characteristics (tradeoffs). This enables us to opportunistically utilize a better algorithm, e.g., at high load a faster, but more expensive algorithm such as a hardware co-processor, which can be utilized in other more urgent analyses at low load. The goal of this research

work is to minimize the impact on the data streams while performing algorithm switching, e.g., the disturbance on output results.

The rest of the paper is organized as follows. In Sect. 2, we discuss the potential problems of the runtime switching among distributed stream processing algorithms. The related work to our research is presented in Sect. 3. In Sect. 4, we describe the research challenges for the algorithm switching. The preliminary solution and result of our research are presented in Sect. 5. Finally, we conduct a research plan for this research work in Sect. 6.

2 Problem Statement

In this section, we start with a plain stream redirection to characterize the potential problems in runtime switching among distributed stream processing algorithms. Finally, we discuss the main goals that we want to achieve in this research work.

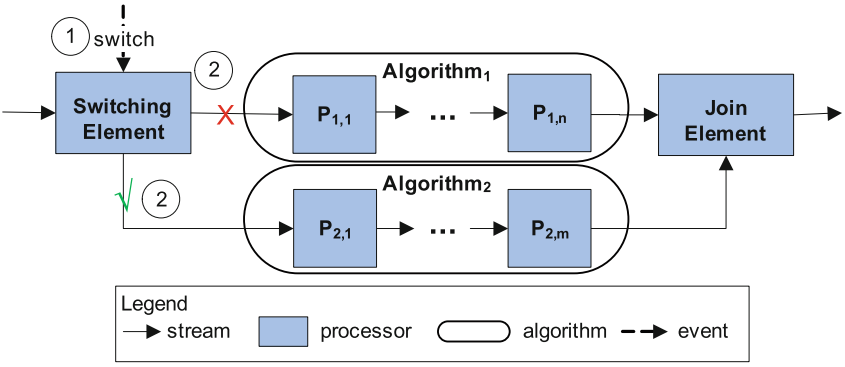


Fig. 1. Plain stream redirection.

Figure 1 depicts plain stream redirection in terms of a data flow diagram, i.e., nodes represent (distributable) data processors and edges the data flow. The distributed stream processing algorithms are presented with several processor nodes. As shown in Fig. 1, the processors $P_{1,1}$ to $P_{1,n}$ constitute *Algorithm*$_1$ ($P_{2,1}$ to $P_{2,m}$ for *Algorithm*$_2$). In addition, two guarding processors control input (*Switching Element*) and output (*Join Element*) streams. Assume that *Algorithm*$_1$ is the currently active algorithm. A signal, i.e., an asynchronous event sent to a processor, indicates the need to switch the active algorithm to another target algorithm ①. In the plain stream redirection, the switch signal causes an immediate re-routing of the data streams ②, i.e., the stream to *Algorithm*$_1$ is disabled and the alternative stream to *Algorithm*$_2$ is enabled at the same time.

In this plain stream redirection, we can potentially see two problems. One is the overlapping results that could be caused by the queuing effects in the

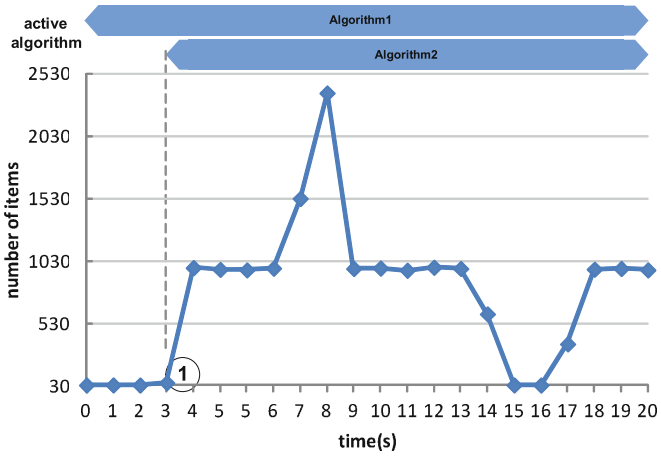


Fig. 2. Throughput caused by the plain stream redirection when switching from a slower algorithm.

processors of the *Algorithm*$_1$ if there is no additional care at the connection between the *Algorithm*$_1$ and the *Join Element* after the switch. Another potential impact is the gap on the processing results, i.e., loss of data queued in the *Algorithm*$_1$ that could occur if we immediately remove the resources on the *Algorithm*$_1$ after switching to *Algorithm*$_2$.

We conduct an evaluation on a cloud-based Storm cluster to verify the impact of the plain stream redirection on data streams. Figure 2 illustrates the throughput (in the *Join Element*) caused by the plain stream redirection when switching from the algorithm with the capacity of 30 items per second (*Algorithm*$_1$) to the one with 1000 items per second (*Algorithm*$_2$). At ①, we send the switch signal and cause a rerouting of the data streams from *Algorithm*$_1$ to *Algorithm*$_2$. As depicted in Fig. 2, the throughput after the switch fluctuates for more than 15 s. To explain this effect, we analyzed the logs written by our implementation of the processors. By comparing the timestamps of the items, we realized that the fluctuations are caused by the overlapping processing in both *Algorithm*$_1$ and *Algorithm*$_2$.

Based on these problems, we consider the following goals for our research work:

G1. Transparency. Our main goal for switching data processing algorithms is to minimize the impact on data streams, i.e., the switch shall happen transparently without disturbance on the processing results. More specifically, we focus on two sub-goals:

(a) **No missing or duplicated data.** Switching an algorithm at runtime must not cause data loss or duplication of data items.

(b) **Minimizing effects on output stream characteristics.** In addition to G1a, also the effect on further (application-) relevant stream characteristics such as latency or throughput shall be minimized.

G2. Minimizing the switching time. This is our secondary goal aiming at switching algorithms at runtime as fast as possible to reduce the time for causing disturbances to the streams.

3 Related Work

Stream processing frameworks including Apache Storm [2] and Apache Spark [1] can be deployed in a Cloud environment to support large-scale data processing. In particular, software systems such as storm-deploy [19] aim at simplifying the deployment of Storm clusters on Cloud offerings including AWS EC2 [4]. Comparing to traditional data processing operators, such cloud-based stream processing frameworks provide high flexibility to realize analysis algorithms, i.e., allow the data analyst to implement own operators (we call *algorithms*), however, they currently do not provide much support for runtime adaptation. There exist **adaptive cloud-based stream processing approaches** in literature. For instance, Satzger et al. [21] present an elastic stream computing platform dynamically attaching and releasing machines to adjust the computational capacities to the current needs. Heinze et al. [15] support auto-scaling of data stream processing to address varying workloads. Similarly, Cervino et al. [8] provide an adaptive approach for dynamically provisioning virtual machines (VMs) based on the current input rates. To enable resource provisioning on different cloud environments, Calheiros et al. [7] propose the Aneka platform to provide a seamless integration of enterprise computing resources (e.g., Desktop Grids and servers) with public Cloud resources while minimizing the costs of utilization of public Cloud resources. Such approaches mainly focus on the adaptation of resource provisioning in the cloud, however they do not address the adaptation on the stream processing algorithms.

There are different **mechanisms to adapt stream processing algorithms** in literature. The typical mechanism is the parameter adaptation of the stream processing algorithm, e.g., dynamically adapting the batch size to improve the performance of the stream processing [12] and adapting the input rate based on the network conditions for media streaming [13]. To cope with varying workloads, Madsen et al. [18] present a dynamic load balancing approach parallelizing operators to scale up the processing. Similar approaches, such as [6,22], automatically extract data parallelism of distributed stream processing. Another type of adaptation for optimizing the performance of stream processing presented by Chatzistergiou et al. [10] is the task reallocation approach reconfiguring stream executing jobs to minimize the transfer latency over cluster-based stream processing systems. To handle overload situations, data admission (also called load shedding) is frequently applied, e.g., the frequency-based load shedding technique proposed by Chang et al. [9] and adaptive input admission control presented in [5]. There are also approaches which adapt the query plan such as Borealis [5] or RTSTREAM [25]. Changing query programs can be seen as a similar approach to our switching of stream processing algorithms, however, they only focus on a fixed set of database-like stream operators. Another similar approach proposed by Hwang et al. [16] switches among active and standby versions

of the same operator to recover from processing errors, but not among alternative algorithms. Probably, the closest approach to ours is by Wei et al. [24], who switch from the efficient just-in-time join strategy to the more powerful structural one for the context-aware XQuery stream processing. However, this approach only focuses on one specific join operator and also does not take the impact into account. Above discussed mechanisms/approaches do support adaptation on stream processing algorithms but none of them focuses on switching among distributed processing algorithms at runtime.

In summary, to our best knowledge, except for our previous published work in [20] there are no approaches on dynamically exchanging alternative stream processing algorithms in the cloud-based environment, which explicitly aim at minimizing the impact on the data streams.

4 Research Challenges

In order to fulfill the requirements discussed in Sect. 2, we identified the following research challenges as part of this research work.

1. **Definition of a "safe" switch.** To avoid the disturbance on the output result of the processing algorithm, we introduce the concept of a "safe" (transparent, gap-free) switch, which takes the characteristics of alternative algorithms into account. We aim at determining a feasible, if not even optimal point in time for switching. This requires us to classify different stream processing algorithms based on their characteristics and to provide different switch mechanisms for different groups of algorithms. The goal of the safe switch is to provide quality guarantees for each group of algorithms, while minimizing the impact on the runtime algorithm adaptation.
2. **Theoretical formalization of the algorithm switch.** To provide the full concept of a generic approach for the algorithm switch, we would need to formalize the algorithm switch based on the "safe" determination as the theoretical part of this research work. This would focus on the theoretical categories of algorithms with different characteristics as well as the theoretical solution of switching among different combinations of algorithms. To perform a "safe" switch, we also need to define the prerequisite for each category of algorithms.
3. **Data tracing.** In order to determine whether the data item is fully processed or still pending for the completion of its processing, we need to trace data items for obtaining such information along with processing nodes of the algorithm. This requires the acknowledgement of each data item to inform its completion. Although recent data stream processing frameworks, such as Apache Storm, support acknowledgement mechanisms, we do not have access to the detailed tracing information.
4. **Data synchronization.** To support achieving the main goal, i.e., the transparency mentioned in the requirement G1, data synchronization on both algorithms is needed to ensure no data loss or duplicated data items before switching to the target algorithm. Depending on the enqueued data items as well

as the synchronization mechanisms, it may require the transferring of data items to the target algorithm.

5. **Queuing control.** As discussed in Sect. 2, queuing effects in the original algorithm could cause inconsistent results while performing an algorithm switch. To cope with such effects, we would need queuing control over algorithms to observe the input data feeding into the respective algorithms. However, the amount of enqueued data items could also have a significant impact on the switching time as we may need to transfer them to the target algorithm during data synchronization. For this case, we would adopt Backpressure [11] or admission control [5].
6. **Dynamic algorithm deployment.** To optimize the resource usage, we must support dynamic deployment of alternative algorithms, i.e., deploying only the selected processing algorithm, avoiding the use of unnecessary processing nodes in the cloud. For achieving such flexibility, the alternative algorithms must support dynamical connection to the current processing.

5 Preliminary Solution and Result

In this section, we present our initial idea (more details can be found in [20]) as well as the preliminary results we achieved so far regarding the research challenges listed in Sect. 4.

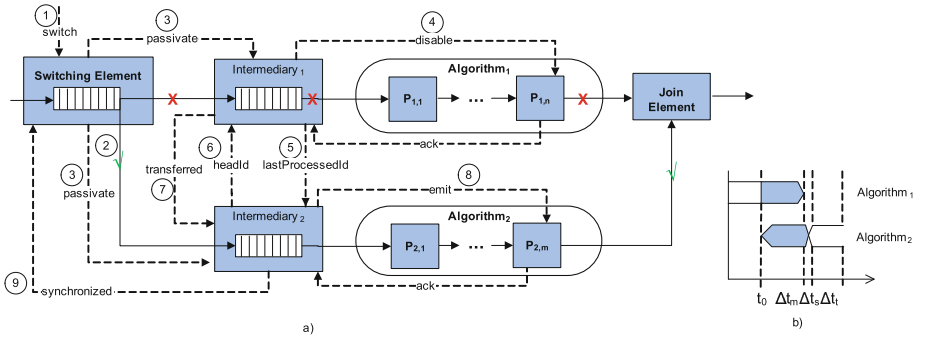


Fig. 3. The preliminary solution of the algorithm switch in terms of (a) signals and (b) timing.

To handle the queuing effects explained in Sect. 2, we introduced an **Entrance Queue** to explicitly control the input data feeding into the respective algorithms. As illustrated in Fig. 3, we represent the entrance queues as individual processors (*Intermediary*$_1$ and *Intermediary*$_2$). The queues in the intermediary processors remove an item only when it is fully processed by the last processing node of the respective algorithm. This is indicated by an acknowledgement signal (*ack* in Fig. 3) sent by $P_{1,n}$ or $P_{2,m}$, i.e., the last node of the

respective algorithm. The items in the queues are either pending to be processed or emitted to the respective algorithm but not fully processed yet. We also utilize a further queue in the *Switching Element* to control the overall stream ingestion.

To maintain the state for the target algorithm, we currently apply a parallel track strategy [17] to run the original and target algorithm for a time Δt_m in parallel to enable the target algorithm for creating and stabilizing its state. Δt_m, the so-called warm-up time depends on the involved algorithms and can, e.g., be the time frame of a sliding window.

Let again *Algorithm₁* be the currently active algorithm and let t_0 be the point in time when *Switching Element* receives the **switch signal** ①. In addition to the overall design depicted in Fig. 3(a), we illustrate the individual phases of the approach in the timing diagram in Fig. 3(b). The switch signal initiates the runtime algorithm switching. First we **warm-up** the target algorithm in parallel to build its state, i.e., activate the stream to *Algorithm₂* by duplicating the input items in the *Switching Element* ②. At the end, *Join Element* passes only the items of the active algorithm, i.e., it discards the output of the passive algorithms, in particular the output of *Algorithm₂* in the warm-up phase. Now, both algorithms process the input stream in parallel for Δt_m.

When the warm-up time is over, the actual switch happens at $t_0 + \Delta t_m$ as indicated in Fig. 3(b) by performing the **data synchronization** between both entrance queues, i.e., *Intermediary₁* negotiates with *Intermediary₂* the last processed item in *Algorithm₁* as a basis for the queue transfer. During switching, each item is queued in the intermediary processors along with a sequential identifier indicating the arrival order of the items. Let *lastProcessedId* be the identifier of the last item emitted to *Algorithm₁*. However, ongoing data processing during the synchronization may invalidate *lastProcessedId*. Therefore, we first passivate ③ both algorithms during the synchronization and disable ④ also the output of results in $P_{1,n}$ to avoid that acknowledgment signals disturb the synchronization. $P_{1,n}$ confirms the passivation (not shown on Fig. 3) so that *Intermediary₁* can now send *lastProcessedId* to *Intermediary₂* ⑤. We denote the time needed for synchronization as Δt_s.

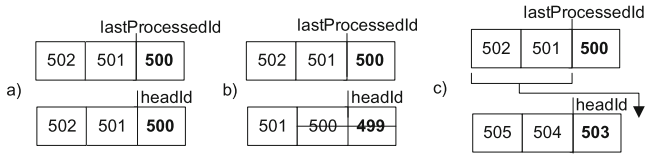


Fig. 4. Three cases of output synchronization.

Let *headId* be the identifier of the head of the queue in *Intermediary₂*. As the involved algorithms may operate at a different speed, we must consider three cases for the synchronization (illustrated as queues for the intermediary processors in Fig. 4):

(a) If $headId = lastProcessedId$, both algorithms are running at the same speed. No items must be transferred and $Algorithm_2$ can immediately take over the processing from $Algorithm_1$ as shown in Fig. 4(a).
(b) If $headId < lastProcessedId$ then $Algorithm_1$ is faster than $Algorithm_2$. In this case, no items must be transferred, but the items [$headId$, $lastProcessedId$] must be skipped as they would cause duplicated results. For example, in Fig. 4(b) items [499, 500] have been processed and must be skipped.
(c) If $headId > lastProcessedId$, items ($lastProcessedId$, $headId$) must be transferred to $Intermediary_2$. Therefore, $Intermediary_2$ sends the $headId$ to $Intermediary_1$ ⑥ and initiates the queue transfer, i.e., $Intermediary_1$ sends unprocessed items via network to $Intermediary_2$. Let Δt_t be the transfer time. At $t_t = t_0 + \Delta t_m + \Delta t_s + \Delta t_t$, $Intermediary_2$ is notified about the end of the queue transfer ⑦ to prepare for regular items from *Switching Element*. In the example in Fig. 4(c), the lower queue is processed faster than the upper queue. So the items (500, 503) must be transferred to avoid a gap.

Due to the *ack* signals from $P_{2,m}$, $Intermediary_2$ can track whether all data items for warm-up have been processed. As soon as synchronized items are passed to $Algorithm_2$, $Intermediary_2$ sends an *emit* signal ⑧ to $P_{2,m}$ enabling the output of processed data to *Join Element*. To minimize the switching time, $Algorithm_2$ processes synchronized items in parallel to the queue transfer, i.e., it starts processing real data already during queue transfer. Finally, $P_{2,m}$ confirms the activation of the output stream (not shown in Fig. 3). In turn, $Intermediary_2$ notifies *Switching Element* about the end of the synchronization ⑨ as well as that $Algorithm_2$ took over the processing and $Algorithm_1$ is discarded.

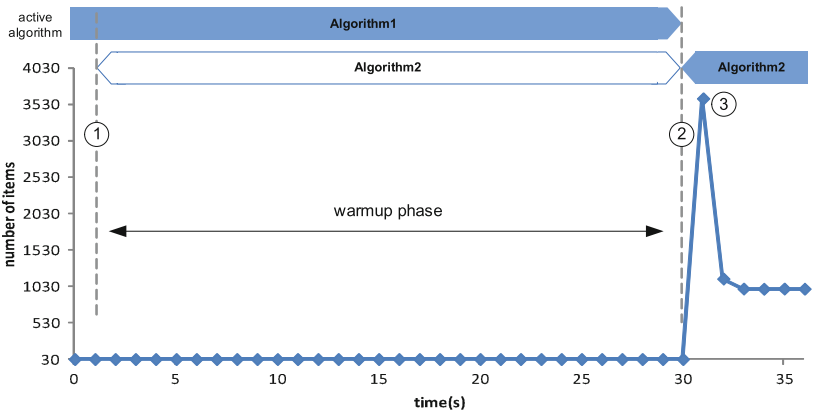


Fig. 5. Throughput of the preliminary solution when switching from a slower algorithm.

As a comparison, we evaluate this initial solution with the same scenario used for the plain stream redirection in Sect. 2. As illustrated in Fig. 5, we switch

from a slower algorithm with a capacity of 30 items per second to one with 1000 items per second. The actual switch signal arrives at ① and both algorithms run in parallel until ②, the end of the warm-up phase of 30 s. During the warm-up phase the output is still produced by the *Algorithm*$_1$. The output synchronization starts at ②. Switching from a slower algorithm, i.e., the third synchronization case, leads to a queue transfer from *Algorithm*$_1$ to *Algorithm*$_2$. *Algorithm*$_2$ starts processing data in parallel to the queue transfer so that the switching time is not dominated by Δt_t. However, several transferred items arrive in short time at *Algorithm*$_2$ and lead to a throughput peak ③. As we analyzed from the logs, in this experiment the actual switching time takes 106 ms, mostly due to the 9 signals (each taking around 10 ms) sent during the switch.

As discussed in Sect. 2, plain stream redirection suffers from queuing effects, item duplication and requires an overall switching time of more than 15 s. In contrast, this initial improved approach reduces the switching time to less then 110 ms without impacting the output accuracy.

6 Research Plan

To fulfill our listed requirements as well as address the mentioned challenges, we present the research plan of this PhD work below by listing the steps that we intend to perform.

Step 1. Conducting a systematic literature review on runtime stream processing adaptation. This step is used to obtain an overview on the runtime adaptation of data stream processing. It aims at gaining insight into different adaptation approaches in order to acquire the state of the art.

Step 2. Formalization of the "safe" algorithm switch. This step aims at formalizing the algorithm switch considering the "safe" concept as the theoretical part of this research work. It mainly contains two substeps. (1) Classification of the stream processing algorithms with different characteristics to be addressed in our algorithm switch approach. (2) Determining the prerequisite of each group of algorithms for performing a "safe" switch.

Step 3. Design of a generic approach for algorithm switch. We aim at designing a generic switch approach which fulfills the requirements for the groups of algorithms formalized in the Step 2. The generic switch approach is intended to apply respective strategies for the algorithms with different characteristics to achieve different levels of quality guarantees.

Step 4. Evaluation of the feasibility of the proposed approach. Before starting to implement the solution, we plan to make an early evaluation of the feasibility of the proposed approach, in particular, to prevent from potential development problems.

Step 5. Design and Implementation of the switch toolbox. The aim of this step is to build a flexible switch toolbox which can easily integrate different switch strategies. This could support the implementation of the proposed switch approach and also benefit the comparison of switch techniques with different characteristics.

Step 6. Evaluation of the generic switch approach. As the last step, we evaluate the implemented switch approach. We mainly focus on two types of evaluation: (a) Technical evaluation, e.g., the characteristics such as latency, throughput, etc. (b) Developer study, e.g., the usability of our approach for developers.

We are currently finalizing Steps 1 and 2. Also, based on the preliminary solution, we started designing a high-level structure of the switch toolbox (Step 5), which enables further detailed implementation of our switch approach.

Acknowledgments. This work is partially supported by the European Commission in the 7th framework programme through the QualiMaster project (grant 619525).

References

1. Apache spark. Lightning-fast cluster computing. http://spark.apache.org/. Accessed 06 Oct 2016
2. Apache storm. Distributed and fault-tolerant realtime computation. http://storm.apache.org/. Accessed 06 Oct 2016
3. Andrade, H.C.M., Gedik, B., Turaga, D.S.: Fundamentals of Stream Processing: Application Design, Systems, and Analytics. Cambridge University Press, Cambridge (2014)
4. Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A., Buyya, R.: Big data computing and clouds: trends and future directions. J. Parallel Distrib. Comput. **79**, 3–15 (2015)
5. Balkesen, C., Tatbul, N., Özsu, M.T.: Adaptive input admission and management for parallel stream processing. In: Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS), pp. 15–26. ACM (2013)
6. Brito, A.: Optimistic parallelization support for event stream processing systems. In: Proceedings of the 5th Middleware Doctoral Symposium, pp. 7–12 (2008)
7. Calheiros, R.N., Vecchiola, C., Karunamoorthy, D., Buyya, R.: The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. Future Gener. Comput. Syst. **28**, 861–870 (2012)
8. Cervino, J., Kalyvianaki, E., Salvachua, J., Pietzuch, P.: Adaptive provisioning of stream processing systems in the cloud. In: 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW), pp. 295–301. IEEE (2012)
9. Chang, J.H., Kum, H.-C.M.: Frequency-based load shedding over a data stream of tuples. Inf. Sci. **179**(21), 3733–3744 (2009)
10. Chatzistergiou, A., Viglas, S.D.: Fast heuristics for near-optimal task allocation in data stream processing over clusters. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM), pp. 1579–1588 (2014)
11. Collins, R.L., Carloni, L.P.: Flexible filters: load balancing through backpressure for stream programs. In: Proceedings of the Seventh ACM International Conference on Embedded Software (EMSOFT), pp. 205–214 (2009)
12. Das, T., Zhong, Y., Stoica, I., Shenker, S.: Adaptive stream processing using dynamic batch sizing. In: Proceedings of the ACM Symposium on Cloud Computing (SOCC), pp. 16:1–16:13 (2014)

13. Goudarzi, H., Salavati, A.H., Pakravan, M.R.: An ant-based rate allocation algorithm for media streaming in peer to peer networks: extension to multiple sessions and dynamic networks. J. Netw. Comput. Appl. **34**(1), 327–340 (2011)
14. Heinze, T., Meyer, P., Jerzak, Z., Fetzer, C.: Measuring and estimating monetary cost for cloud-based data stream processing (demo). In: Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS), pp. 333–334 (2013)
15. Heinze, T., Pappalardo, V., Jerzak, Z., Fetzer, C.: Auto-scaling techniques for elastic data stream processing. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS), pp. 318–321 (2014)
16. Hwang, J.-H., Balazinska, M., Rasin, A., Çetintemel, U., Stonebraker, M., Zdonik, S.: High-availability algorithms for distributed stream processing. In: International Conference on Data Engineering (ICDE), pp. 779–790 (2005)
17. Rundensteiner, E.A., Ding, L., Zhu, Y., Sutherland, T.M., Pielech, B.: CAPE: a constraint-aware adaptive stream processing engine. Stream Data Manag. **30**, 83–111 (2005). Springer
18. Madsen, K.G.S., Zhou, Y.: Dynamic resource management in a massively parallel stream processing engine. In: Proceedings of the 24th ACM International on Information and Knowledge Management, pp. 13–22 (2015)
19. Marz, N.: Storm-deploy. https://github.com/nathanmarz/storm-deploy/. Accessed 06 Oct 2016
20. Qin, C., Eichelberger, H.: Impact-minimizing runtime switching of distributed stream processing algorithms. In: Big Data Processing - Reloaded Workshop of the EDBT/ICDT Joint Conference (2016)
21. Satzger, B., Hummer, W., Leitner, P., Dustdar, S.: ESC: towards an elastic stream computing platform for the cloud. In: 4th IEEE International Conference on Cloud Computing (CLOUD), pp. 348–355 (2011)
22. Schneider, S., Hirzel, M., Gedik, B., Wu, K.-L.: Auto-parallelizing stateful distributed streaming applications. In: Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, pp. 53–64 (2012)
23. Vijayakumar, S., Zhu, Q., Agrawal, G.: Dynamic resource provisioning for data streaming applications in a cloud environment. In: Proceedings of the 2nd Cloud Computing Technology and Science (CloudCom), pp. 441–448 (2010)
24. Wei, M., Rundensteiner, E.A., Mani, M., Li, M.: Processing recursive xquery over xml streams: the raindrop approach. Data Knowl. Eng. **65**(2), 243–265 (2008)
25. Wei, Y., Son, S.H., Stankovic, J.A.: RTSTREAM: real-time query processing for data streams. In: International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2006), pp. 141–150 (2006)

Abstracts

Cloudiator – Enacting Deployments and Adaptation in PaaSage

Daniel Baur(✉) and Jörg Domaschka

Institute of Information Resource Management, University of Ulm,
Albert-Einstein-Allee 43, 89081 Ulm, Germany
{daniel.baur,joerg.domaschka}@uni-ulm.de
http://www.uni-ulm.de/in/omi

The EU FP7 PaaSage project (http://www.paasage.eu/) realises a model-based cross-cloud development and deployment platform easing the access and management of cloud resources for its users (developers and operators). For this task, PaaSage leverages upon a three phase workflow consisting of *modelling*, *deployment* and *execution*. During the *modelling phase* the user creates a cloud-provider independent model using the domain specific Cloud Application Modelling and Execution Language (CAMEL). During the *deployment phase* a constraint problem is created by matching the user's model with available cloud provider profiles. The Reasoner component solves the constraint problem, resulting in a cloud-provider dependant model, depicting the offers to use for the deployment (possibly from different cloud providers). Subsequently, a deployment plan is calculated by the Adapter, specifying the single actions to be executed to deploy the model. Finally, during the *execution phase*, the deployment plan is executed by the Executionware. A feedback loop using monitoring data from the Executionware is used to improve future deployments.

Cloudiator [1] (https://github.com/cloudiator) is a cross-cloud orchestration and management tool, that acts as the Executionware of PaaSage. Thus, it provides the following features: *(i)* it supports cross-cloud deployments meaning that it can place different component instances of a single application across multiple clouds; *(ii)* it is multi-tenant capable, meaning that it supports multiple applications and users at a time; *(iii)* it provides an adaptive monitoring & aggregation solution and *(iv)* it provides a runtime management capable of automatically addressing adaptation actions like scaling.

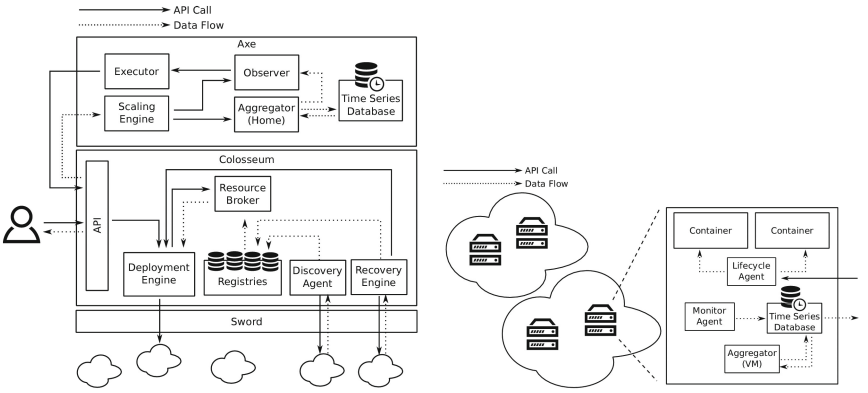
For these tasks it relies on a distributed architecture as depicted in Fig. 1. Its components are split into two domains, the cloud domain running on acquired VMs in the cloud and the home domain located on the PaaSage installation of the user. The workflow starts with the input of the deployment plan, in PaaSage's case provided by the Adapter. This plan contains the exact configuration for the virtual machines (image, hardware, location) as well as the component placement. In addition the description defines the monitoring demands, accompanied with a set of scaling rules used for addressing runtime management. Based on the deployment plan, the deployment engine will allocate the defined virtual machines, using our abstraction layer Sword, that is hiding semantical and syntactical differences in the cloud providers' APIs. Afterwards, it instructs Lance to deploy the components by executing the interface actions (e.g. shell-scripts)

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 289–291, 2018.
https://doi.org/10.1007/978-3-319-72125-5

defined in the CAMEL model. It uses an imperative workflow for executing the actions, that is derived from defined communication dependencies in the model [3].

Afterwards, the application's runtime behaviour is monitored by our monitoring agent Visor, offering the possibility to install multiple sensors and reconfigure them during runtime. The monitoring data is fed into the Axe [2] system responsible for aggregation and derivation of composite metrics. For this purpose it uses a hierarchical, distributed system of aggregators aiming to reduce the communication effort. Finally, if measurements violate the thresholds defined in the scaling rules, Axe will enact the user-defined mitigation actions.

In addition, Cloudiator provides the discovery of cloud provider offers, that are being fed into PaaSage's provider profiles and a watchdog and recovery engine, addressing common failures in the deployment (like unresponsive VMs).



(a) Home Domain (b) Cloud Domain

Fig. 1. Cloudiator's Architecture

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 317715 (PaaSage).

References

1. Baur, D., Domaschka, J.: Experiences from building a cross-cloud orchestration tool. In: Proceedings of the 3rd Workshop on CrossCloud Infrastructures and Platforms, CrossCloud 2016, pp. 4:1–4:6 (2016)
2. Domaschka, J., Seybold, D., Griesinger, F., Baur, D.: Axe: a novel approach for generic, flexible, and comprehensive monitoring and adaptation of cross-cloud applications. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 184–196. Springer, Cham (2016)

3. Domaschka, J., Griesinger, F., Baur, D., Rossini, A.: Beyond mereapplication structure thoughts on the future of cloud orchestration tools. Procedia Comput. Sci. **68**, 151–162 (2015). 1st International Conference on Cloud Forward: From Distributed to Complete Computing

BPaaS Execution in CloudSocket

Daniel Seybold[1(✉)], Robert Woitsch[2], Jörg Domaschka[1], and Stefan Wesner[1]

[1] Institute of Information Resource Management, Ulm University, Ulm, Germany
daniel.seybold@uni-ulm.de
[2] BOC Asset Management GmbH, Vienna, Austria
robert.woitsch@boc-eu.com

Abstract. The H2020 research CloudSocket project enacts the business IT-alignment by implementing Business process as a Service (BPaaS).

1 Introduction

Cloud Computing has a still growing influence on the IT ecosystem, especially for business applications. Still challenging is the so-called *business and IT alignment (BITA)*. BITA bridges the gap between the business and IT domains and ensures sufficient and the right hardware/software resources are available to handle a company's business processes. We consider BITA as one of the key success factors for cloud usage. For supporting BITA in clouds, the current application centric view needs to be enhanced with a corresponding *business process (BP)* view. The CloudSocket project[1] enables such a BP-oriented view on cloud offerings and supports cloud-enabled BITA. In particular, it implements BITA by offering a new cloud service level: *Business process as a Service (BPaaS)* [3].

CloudSocket provides a brokerage platform for BPaaS. A CloudSocket Broker plans and designs the BPs, implements executable *workflows (WFs)* and creates *BPaaS Bundles*. The BPaaS Bundles are deployable in the cloud and offered by a CloudSocket Broker to its customers through a marketplace.

2 CloudSocket Architecture

The CloudSocket architecture builds upon the four loosely coupled buildings blocks shown in Fig. 1.

The *Design Environment* creates the BPaaS Design Package by editing domain specific BP models and executable WF models, mapping a domain specific BP model to an executable WF model as well as semantically annotating BP and WF models. The Design Environment is based on ADONIS[2].

The *Allocation Environment* enriches the executable WF of the BPaaS Design Package with concrete and deployable cloud services. The cloud services are modelled in CAMEL [2] and included in the BPaaS Bundle. Design and Allocation Environment compose a management environment assisting the broker in managing the BPaaS offerings.

[1] https://www.cloudsocket.eu/.
[2] http://en.adonis-community.com/.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 292–293, 2018.
https://doi.org/10.1007/978-3-319-72125-5

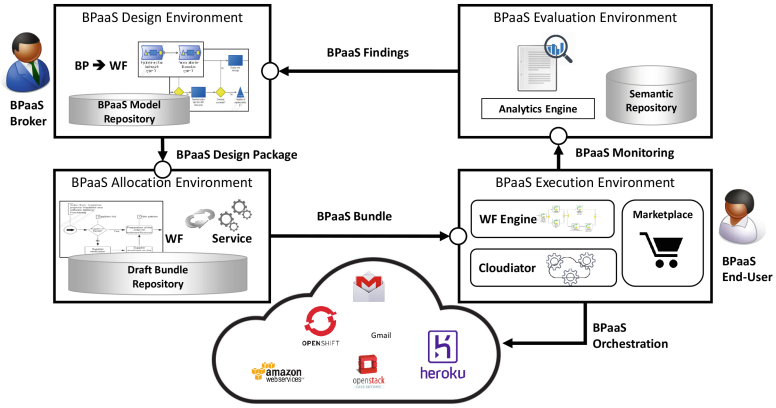


Fig. 1. The CloudSocket architecture

The *Execution Environment* offers BPaaS Bundles via a marketplace to End-Users. As a Bundle is purchased, Cloudiator [1] orchestrates, monitors and adapts the cloud services. End-Users interact with the services via the WF Engine[3]. Based on the semantic enrichment in the management environments, the Execution Environment can operate on a semantically enriched BPaaS Bundle.

The *Evaluation Environment* provides optimisation suggestions (KPI assessment, optimised BPaaS deployment and adaptation patterns) to the Broker by abstracting technical data back to domain specific business decision.

Currently CloudSocket reaches the first half of the project with the release of the 1st prototype[4], enabling the holistic BPaaS life cycle.

Acknowledgements. The research leading to these results has received funding from the European Community's Framework Programme for Research and Innovation HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

1. Domaschka, J., Baur, D., Seybold, D., Griesinger, F.: Cloudiator: a cross-cloud, multi-tenant deployment and runtime engine. In: 9th SummerSOC (2015)
2. Kritikos, K., Domaschka, J., Rossini, A.: SRL: a scalability rule language for multi-cloud environments. In: IEEE 6th International Conference on, CloudCom 2014, pp. 1–9. IEEE (2014)
3. Woitsch, R., Utz, W.: Business Processes as a Service (BPaaS): a model-based approach to align business with cloud offerings. In: eChallenges e-2015 Conference, pp. 1–8. IEEE (2015)

[3] https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Workflow+Engine+Component.

[4] https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Components.

Context-Aware Cloud Topology Optimization for OpenStack

Christopher B. Hauser(✉), Athanasios Tsitsipas, and Jörg Domaschka

Institute of Information Resource Management, Ulm University, Ulm, Germany
{Christopher.Hauser,Athanasios.Tsitsipas,joerg.domaschka}@uni-ulm.de

Abstract. CACTOS offers Cloud developers, operators, and consultants a context-aware optimisation for private Clouds. It leads to better and more reliable user experience, by optimising the mapping of virtual to physical resources, considering application requirements and heterogeneity. The optimisation and simulation requires monitoring, and an integration for controlling and intercepting client requests.

Keywords: Cloud data centres · Optimization · Workload placement

1 Why CACTOS?

CACTOS'[1] overall goal is to optimise a Cloud data centre from infrastructure to application level for better performance, better utilisation and a higher energy efficiency [1]. CACTOS therefore extensively monitors on infrastructure level, analyses application profiles and enhances the functionality of off the shelf Cloud middleware OpenStack (cf. Fig. 1) with improved scheduling and workload reallocation algorithms.

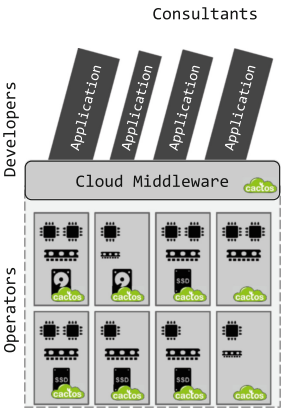


Fig. 1. A CACTOS cloud

For Cloud *developers* CACTOS aims at better experience, and more reliability due to a requirement aware resource scheduling. Since CACTOS is context-aware, the hardware affinity of applications is considered as well as a heterogeneous infrastructure. Cloud data centre *operators* benefit from an increased overall utilisation and hence cost and energy efficiency. Operators can choose between load balancing or consolidating virtual resources on physical hardware. CACTOS continously optimises the placement of applications in a heterogeneous resource pool. CACTOS enables consultants with the capability to detect bottlenecks on hard- and software level. Either manually or automatically via CACTOS reorganising application and hardware infrastructure can be simulated and then enacted for overall improvements.

[1] http://www.cactosfp7.eu.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 294–295, 2018.
https://doi.org/10.1007/978-3-319-72125-5

2 CACTOS Runtime Toolkit in OpenStack

To enable CACTOS in an OpenStack cloud, the monitoring and analysis needs to be connected with the data centre. Bidirectional interactions from the Cloud middleware to CACTOS and back need to be established. Finally, CACTOS can optimise and simulate the OpenStack data centre (cf. Fig. 2).

Data centre analysis starts with monitoring on each host of the data centre. The Chukwabased tool gets detailed information from physical and virtual level. The data is processed and stored in HBase. The huge amount of data is used to create application profiles.

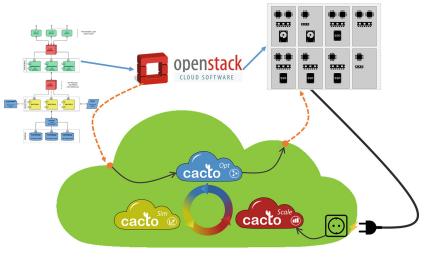


Fig. 2. CACTOS Toolkit Openstack

Invocations to the OpenStack REST API are intercepted by a CACTOS-aware HTTP proxy. The proxy delegates calls for creating and deleting VMs synchronously to CACTOS. For periodical optimisations, CACTOS accesses OpenStack to migrate VMs and via an IPMI Proxy controls the power state of hosts. CACTOS hence controls the workload placement for existing and new VMs.

CACTOS gets the virtual and physical infrastructure with load information from the HBase. Depending on configuration and utilisation values, CACTOS optimises for e.g. load balancing or consolidation. Additionally, an event-driven simulation uses HBase to predict the data centre workload behaviour.

3 Project Status and Results

The recent CACTOS results are publicly available and linked on the website. The tools will be available on GitHub[2] with the project ending in 2016.

Acknowledgment. This research was funded by European Commission's FP7 (grant agrmt. 610711).

Reference

1. Ostberg, P.-O., et al.: The CACTOS vision of context-aware cloud topology optimization and simulation. In: IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014, pp. 26–31 (2014)

[2] https://github.com/cactos.

Envisage: Developing SLA-Aware Deployed Services with Formal Methods

Elvira Albert[1(✉)], Frank de Boer[2], Reiner Hähnle[3], Einar Broch Johnsen[4], and Cosimo Laneve[5]

[1] Complutense University of Madrid, Madrid, Spain
elvira@fdi.ucm.es
[2] CWI Amsterdam, Amsterdam, The Netherlands
f.s.de.boer@cwi.nl
[3] TU Darmstadt, Darmstadt, Germany
haehnle@cs.tu-darmstadt.de
[4] University of Oslo, Oslo, Norway
einarj@ifi.uio.no
[5] University of Bologna – INRIA FOCUS, Bologna, Italy
cosimo.laneve@unibo.it

Insufficient scalability and bad resource management of software services can easily eat up any potential savings from cloud deployment. Failed service-level agreements (SLAs) cause penalties for the provider, while oversized SLAs waste resources on the customer's side. IBM Systems Sciences Institute estimates that a defect which costs one unit to fix in design, costs 15 units to fix in testing (system/acceptance) and 100 units or more to fix in production [6]; this cost estimation does not even consider the *impact cost* due to, for example, delayed time to market, lost revenue, lost customers, and bad public relations. The Envisage project aims at shifting deployment decisions from the end of the software engineering process to become an integral part of software design [2].

Deployment on the cloud gives software designers far reaching control over the resource parameters of the execution environment, such as the number and kind of processors, the amount of memory and storage capacity, and the bandwidth. In this context, designers can also control their software's trade-offs between the incurred cost and the delivered quality-of-service. SLA-aware services, which are *designed for scalability*, can even change these parameters dynamically, at runtime, to meet their service contracts. Envisage permits to design and validate these services by connecting executable models to formal service contracts and an API that is an abstraction of the cloud environment, see Fig. 1. This approach enables new kinds of analysis:

- **Simulation ("Early modeling"):** The formally defined modeling language ABS [10] realizes a separation of concerns between the *cost* of execution and the *capacity* of dynamically provisioned cloud resources [11]. Models are executable; a simulation tool supports rapid prototyping and visualization.
- **Formal methods ("Early analysis"):** as ABS was designed for analysis, it enables a range of tool-supported formal techniques, including behavioral

Supported by the EU project FP7-610582 *Envisage: Engineering Virtualized Services* (http://www.envisage-project.eu).

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 296–298, 2018.
https://doi.org/10.1007/978-3-319-72125-5

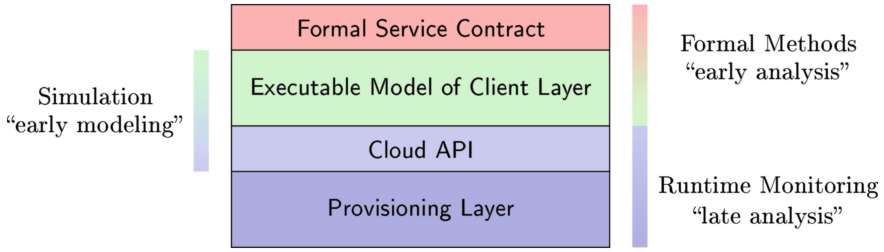


Fig. 1. Making services SLA-aware by means of formal methods, from [9].

types for deadlock analysis and SLA compliance [8], worst-case cost analysis [1], deductive verification [7], and automated test-case generation [4].

– **Monitoring ("Late analysis"):** ABS supports code generation backends [5] that preserve upper bounds on cost and permit performance monitoring of the provisioned cloud resources after deployment [13].

The modeling approach and analyses developed in Envisage have been successfully applied in an industrial context to SDL Fredhopper's eCommerce Optimization [3] and Apache Hadoop YARN [12].

References

1. Albert, E., Arenas, P., Flores-Montoya, A., Genaim, S., Gómez-Zamalloa, M., Martin-Martin, E., Puebla, G., Román-Díez, G.: SACO: Static analyzer for concurrent objects. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 562–567. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_46
2. Albert, E., de Boer, F., Hähnle, R., Johnsen, E.B., Laneve, C.: Engineering virtualized services. In: 2nd Nordic Symposium on Cloud Computing & Internet Technologies. NordiCloud 2013, pp. 59–63. ACM Press (2013)
3. Albert, E., de Boer, F.S., Hähnle, R., Johnsen, E.B., Schlatte, R., Tapia Tarifa, S.L., wong, P.Y.H.: Formal modeling and analysis of resource management for cloud architectures: an industrial case study using real-time ABS. J. Service-Oriented Comput. Appl. **8**(4), 323–339 (2014)
4. Albert, E., Gómez-Zamalloa, M., Isabel, M.: SYCO: a systematic testing tool for concurrent objects. In: Compiler Construction, CC 2016. ACM (2016)
5. Bezirgiannis, N., de Boer, F.: ABS: a high-level modeling language for cloud-aware programming. In: Freivalds, R., Engels, G., Catania, B. (eds.) SOFSEM 2016. LNCS, vol. 9587, pp. 433–444. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49192-8_35
6. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IEEE Trans. Softw. Eng. **14**(10), 1462–1477 (1988)
7. Din, C.C., Bubel, R., Hähnle, R.: KeY-ABS: a deductive verification tool for the concurrent modelling language ABS. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 517–526. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_35

8. Giachino, E., Laneve, C., Lienhardt, M.: A framework for deadlock detection in ABS. Softw. Syst. Model. (2016, To Appear)
9. Hähnle, R., Johnsen, E.B.: Designing resource-aware cloud applications. IEEE Comput. **48**(6), 72–75 (2015)
10. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: a core language for abstract behavioral specification. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) FMCO 2010. LNCS, vol. 6957, pp. 142–164. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25271-6_8
11. Johnsen, E.B., Schlatte, R., Tapia Tarifa, S.L.: Integrating deployment architectures and resource consumption in timed object-oriented models. J. Logical Algebraic Methods Program. **84**(1), 67–91 (2015)
12. Lin, J.-C., Yu, I.C., Johnsen, E.B., Lee, M.-C.: ABS-YARN: a formal framework for modeling hadoop yarn clusters. In: Stevens, P., Wasowski, A. (eds.) FASE 2016. LNCS, vol. 9633, pp. 49–65. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49665-7_4
13. Nobakht, B., de Gouw, S., de Boer, F.S.: Formal verification of service level agreements through distributed monitoring. In: Dustdar, S., Leymann, F., Villari, M. (eds.) ESOCC 2015. LNCS, vol. 9306, pp. 125–140. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24072-5_9

Molecular Dynamics with HyperFlow and Scalarm on the PaaSage Platform

Maciej Malawski[1(✉)], Bartosz Balis[1], Kamil Figiela[1], Maciej Pawlik[1], Marian Bubak[1], Dariusz Król[1], Renata Słota[1], Michał Orzechowski[1], Jacek Kitowski[1], and Dennis Hoppe[2]

[1] Department of Computer Science, Faculty of Computer Science, Electronics, and Telecommunications, AGH University of Science and Technology, Mickiewicza 30, 30-059 Krakow, Poland
{malawski,dkrol}@agh.edu.pl
[2] High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Nobelstr. 19, 70569 Stuttgart, Germany
dennis.hoppe@hlrs.de

This paper demonstrates how scientific workflow applications executed by HyperFlow engine [1] and data farming experiments managed by the Scalarm platform [2] can benefit from the capabilities offered by PaaSage platform[1], on the example of molecular dynamics simulation. While HyperFlow engine enables users to execute tasks of scientific workflows and Scalarm supports parameter studies oriented on data farming on available computing resources (e.g. virtual machines in a cloud), the role of PaaSage platform is to provision these resources, deploy an application, and automatically scale them according to the application demands. PaaSage uses a model-based approach, which means that the cloud application together with its requirements needs to be described using Cloud Application Modeling and Execution Language (CAMEL) [4].

The main new capabilities resulting from combining HyperFlow and Scalarm with PaaSage are to deploy the entire runtime environment as part of the application. Consequently, the whole platform such as HyperFlow or Scalarm, supporting a class of applications, can be seen as a single cloud application from the perspective of PaaSage. In particular, the workflow enactment engine, HyperFlow [1], acts as a component of the PaaSage application. Similarly, in the case of the Scalarm platform [2], we modelled the whole platform as an application, which is capable to execute different simulation models and scale itself in a cross-cloud environment. Consequently, we avoid tight coupling to a particular cloud infrastructure and middleware in both cases.

According to the PaaSage methodology, we modelled both HyperFlow and Scalarm using CAMEL. HyperFlow deployment model includes the master node which contains the workflow engine components, while the worker (which is scalable from 1 to n nodes) includes the executor together with application dependencies such as MPI or POV-Ray. Additionally, we deploy Redis and RabbitMQ components, as well as InfluxDB and Grafana as monitoring dashboard. Scalarm's architecture also follows the master-worker design pattern, where the

[1] http://www.paasage.eu.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 299–300, 2018.
https://doi.org/10.1007/978-3-319-72125-5

master part consists of loosely coupled services and is responsible for coordinating experiments, while the worker part handles actual computations. In addition, Scalarm can be run on different clouds as described in [3], hence its actual runtime deployment can be divided vertically and horizontally, where the amount of resources for each part can be dynamically adjusted to the actual load.

We demonstrate the benefits of using HyperFlow and Scalarm with PaaSage by deploying a molecular dynamics (MD) simulation. MD simulations are highly representive for e-science applications, because they comprise resource– and compute–intensive calculations that are massively parallelizable via MPI or OpenMP. These kinds of simulations provide information about how a given substance behaves under a given set of physical conditions including temperature and pressure. It allows us to predict material behaviour, for example for industrial purposes, such as the crack distribution across bridges. Simulations usually extend over multiple iterations starting with a coarse simulation over selected data points, and ending in a fine-granular simulation around a point of interest such as the origin of a crack. That's why accurate simulations require to be executed up to several hundred times with different parameter sets to yield accurate results. Our simulation includes a post-processing step that generates a video out of the raw data received from the simulation.

HPC is usually first-choice when it comes to executing e-science applications. However, the trend is towards a hybrid HPC/Cloud model, where high-performance resources are combined with the advantages of the cloud: flexibility, high availability, and disaster recovery to name but a few. Applications that will benefit from such a hybrid model are, in particular, e-science applications that usually include pre- and post-processing steps such as generating a video, which are not compute intensive. Those tasks can then be moved into the cloud, whereas compute-intensive tasks will continue to run on HPC infrastructure. PaaSage enables us to model our e-science applications once using CAMEL, and then deploy individual tasks, managed by HyperFlow or Scalarm, on different HPC/cloud infrastructures with ease.

Acknowledgments. We thankfully acknowledge the support of the EU FP7-ICT project PaaSage (317715), Polish grant 3033/7PR/2014/2 and AGH grant 11.11.230.124. Access to EC2 Cloud was provided by AWS in Education Grant.

References

1. Balis, B.: HyperFlow: a model of computation, programming approach and enactment engine for complex distributed workows. Future Gener. Comput. Syst. **55**, 147–162 (2016)
2. Król, D., Kitowski, J.: Self-scalable services in service oriented software for cost-effective data farming. Future Gener. Comput. Syst. **54**, 1–15 (2016)
3. Krol, D., Slota, R., Kitowski, J., Dutka, L., Liput, J.: Data farming on heterogeneous clouds. In: 2014 IEEE 7th International Conference on Cloud Computing, pp. 873–880. IEEE, June 2014
4. Nikolov, N., Rossini, A., Kritikos, K.: Integration of DSLs and migration of models: a case study in the cloud computing domain. Procedia Comput. Sci. **68**, 53–66 (2015)

Quality-Aware Development of Big Data Applications with DICE

Giuliano Casale[1(✉)], Elisabetta Di Nitto[2], Pooyan Jamshidi[1], and Damian A. Tamburri[2]

[1] Imperial College London, London, UK
g.casale@imperial.ac.uk
[2] Politecnico di Milano, Milan, Italy

Abstract. The DICE EU H2020 Project is exploring model-driven ways to support the continuous design and development of Quality-aware Data-Intensive applications (DIA) by means of DIA-specific quality analyses (e.g., performance, reliability, cost) applied on key DIA technologies (e.g., Hadoop, Storm or Spark). We outline the DICE toolchain and the challenges it addresses.

1 The DICE Project Explained

Today, most organizations face high market pressure, and their ICT is struggling to accelerate service delivery while preserving production and operations quality. On the one hand, ICT operators lack understanding of the application internals (e.g., architecture, design decisions). On the other hand, development teams are not aware of operation details (e.g., infrastructure topology and its limitations). These issues become critical for Big Data [1]. In fact, the big data domain has seen the rapid growth of interest, e.g., featuring increased use of typical big data technologies, e.g., Hadoop/MapReduce, NoSQL or stream processing. However, the time to market and the cost of ownership of such applications are high.

In this context, DevOps practices address the common isolation between Development and Operations [2]. The main goal of DevOps is to achieve a better quality and continuous DIA architecting by eliminating silos and integrating 'Dev' and 'Ops' activities, with people sharing the same goals and working closely with shared modelling, analysis, development and operations facilities. DICE aims at incorporating within the DevOps movement a model-driven approach that facilitates flow of information from Dev to Ops and enables operations monitoring and anomaly detection capabilities to facilitate flow of information from Ops to Dev. Moreover, DICE, in the form of a compact Eclipse Rich Client Platform[1] (RCP), offers three key tenets in the context of DevOps.

First, DICE enables the data-intensive application design with proper UML annotations (i.e., an Eclipse Papyrus UML profile - see mid-left of Fig. 1) that

This paper has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869.

[1] https://eclipse.org/community/rcpos.php.

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 301–302, 2018.
https://doi.org/10.1007/978-3-319-72125-5

(a) facilitate design-time simulation and optimizations and (b) enables 'Dev' and 'Ops' to work side by side on continuous application architecting using simulation and optimization tools (see other Eclipse RCP plugins in lefft-hand side of Fig. 1) that offers refactoring of architectural designs.

Second, DICE includes analysis tools that exploit monitoring data on running components (i.e., see lower-right of Fig. 1) to detect anomalies and to optimize the configurations of applications (see top middle of Fig. 1, Anomaly detection trace-checking and enhancement tools), as well as feedback loop tools that exploit the same data to offer feedback to the architectural design to identify bottlenecks and continuously refactor DIAs.

Third, DICE offers tools which, based on the "Topology and Orchestration Specification for Cloud Applications" (TOSCA)[2], allow for automatic DIA deployment blueprint generation (see the Repository and Continuous Integration component (CI) in the mid-bottom of Fig. 1) as well as continuous integration thereby allowing automated delivery while facilitating the integration between design-time and runtime.

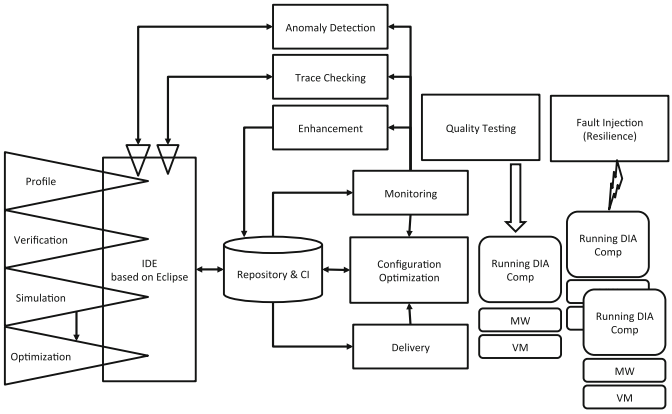


Fig. 1. DICE Architecture, an overview, bottom-left shows Eclipse plugins while in mid-figure appear the integrated components.

Figure 1 provides an overview of the DICE architecture rotating around an Eclipse RCP (bottom-left) while integrated components appear in mid-figure. For more information please refer to deliverable D1.3 in the DICE homepage[3].

References

1. Casale, G., et al.: Dice: quality-driven development of data-intensive cloud applications. In: Proceedings of the 7th International Workshop on Modelling in Software Engineering (MiSE), May 2015
2. Nitto, E.D., et al.: A software architecture framework for quality-aware devops. In: Proceedings of the 2th International Workshop on Quality-Aware DevOps (QUDOS), July 2016

[2] https://www.oasis-open.org/apps/org/workgroup/tosca/members/roster.php.
[3] http://dice-h2020.eu/.

The HORSE Project: IoT and Cloud Solutions for Dynamic Manufacturing Processes

Irene Vanderfeesten(✉), Jonnro Erasmus, and Paul Grefen

School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
{i.t.p.vanderfeesten, j.erasmus, p.w.p.j.grefen}@tue.nl

1 Introduction

During the various stages in the advancement of manufacturing, the pervasiveness of automated equipment gradually increased. Dedicated manufacturing lines improved efficiency and flexible manufacturing systems provided the ability to produce a large variety of products. Reconfigurable manufacturing tools made it possible to combine large volume with improved flexibility between batches [1]. Recently, the demand for customisation is outpacing the ability of manufacturing systems to remain cost effective [2]. The rapid change is forcing factories to become more flexible, while maintaining high volume and quality. The extent of the customisation is also increasing, with products differing based on composition, size, shape, performance or surface treatment. Producing variable products requires a wide range of production activities.

Small and medium enterprises (SME) can't afford to acquire and operate a large number of automated actors to perform numerous activities. They must use their limited personnel and financial capital wisely to compete effectively in the global manufacturing sector. They seek autonomous actors that react to changing variables, based on accurate information processed by a centralised, cloud-based information system. It is recognised that the knowledge of actors and management is now the most valuable asset of a factory, yet enterprise-wide, real-time information and control is still comparatively limited in the manufacturing sector [3].

2 The HORSE Project

The HORSE Project (www.horse-project.eu) is part of the Factories of the Future area of the Horizon 2020 research and innovation programme. The project aims to develop and integrate advanced manufacturing technology in a package accessible to SMEs. The package will include technology such as situational awareness, augmented reality and event-driven process management. To ensure industrial relevance and demonstrate validity, the project consortium includes three commercial manufacturing partners with diverse production activities, e.g. cold forming, casting and assembly.

The proposed technologies are built around a cloud-based information system that will provide centralised control, by aggregating information and invoking the services

A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 303–304, 2018.
https://doi.org/10.1007/978-3-319-72125-5

of actors connected through the Internet-of-Things. This system will make it possible to orchestrate cross-organisational processes, enhancing the potential for customised products by linking the manufacturing activities of multiple enterprises.

Figure 1 shows an overview of the software architecture of the information system. The upper layer represents the centralised orchestration, aggregating data from the shop floor and other information systems to visually inform the user and exert control over the end-to-end manufacturing process. The bottom layer provides the functionality to configure and control individual actors in the manufacturing system, while ensuring safety of humans and equipment.

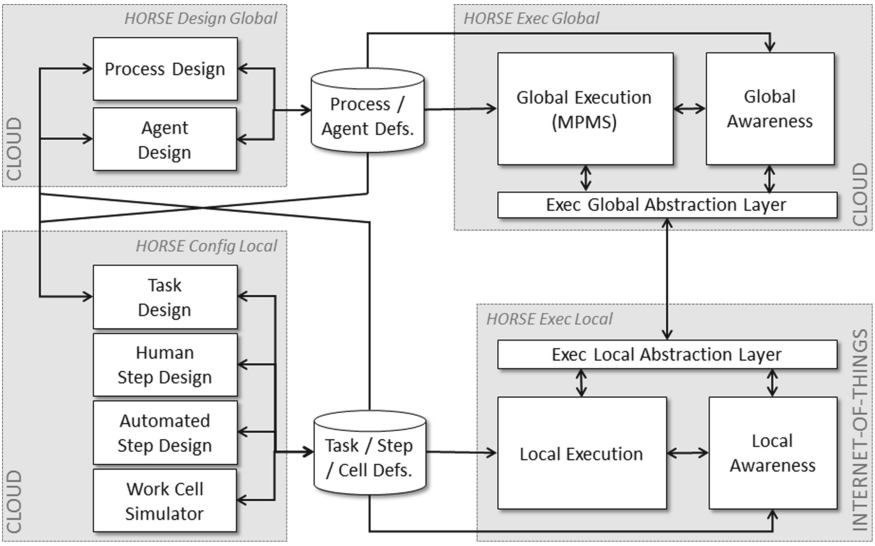


Fig. 1. Software aspect of the HORSE system architecture

Additionally, Fig. 1 shows portions of the architecture envisaged to be enabled by a cloud-based platform, as it contains complex system functionality without strict timing constraints. The bottom-right portion is the connection between the centralised system and the individual actors, with the execution and awareness modules corresponding to the actuating and sensing capabilities of the Internet-of-Things, respectively.

References

1. Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Brussel, H.V.: Reconfigurable manufacturing systems. CIRP Annals-Manufact. Technol. **48**(2), 527–540 (1999)
2. Zhang, Q., Cao, M.: Business process reengineering for flexibility and innovation in manufacturing. Ind. Manage. Data Syst. **102**(3), 146–152 (2002)
3. Zhang, L., Luo, Y., Tao, F., Li, B.H., Ren, L., Zhang, X., Guo, H., Cheng, Y., Hu, A., Liu, Y.: Cloud manufacturing: a new manufacturing paradigm. Enterp. Inf. Syst. **8**(2), 167–187 (2014)

BEACON Project: Software Defined Security Service Function Chaining in Federated Clouds

Massimo Villari[1(✉)], Giuseppe Tricomi[1], Anna Levin[2], Sébastien Dupont[3], and Philippe Massonet[3]

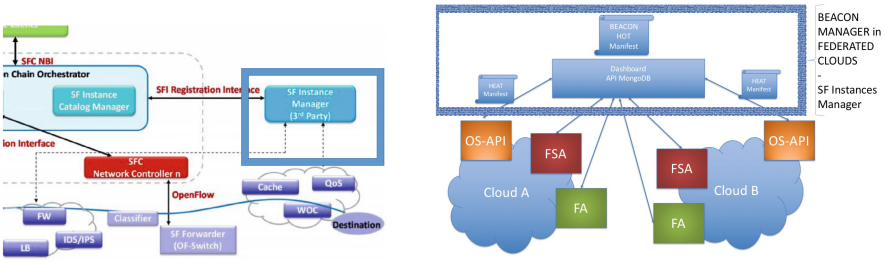
[1] Dep. Ingegneria, University of Messina, Messina, Italy
{mvillari,gtricomi}@unime.it
[2] IBM Research - Haifa, Haifa, Israel
lanna@il.ibm.com
[3] CETIC Research Center, Charleroi, Belgium
{sebastien.dupon,philippe.massonet}@cetic.be

1 Introduction

BEACON project aims at federating cloud networks leveraging virtualised overlay networks over different cloud platforms. The federation of cloud providers is challenging due to the complex interaction between different administrative domains. In this context, the setup of cross cloud networks is compelling, but enabling Service Function Chaining (SFC) is much more complicated. BEACON project uses software-defined approach for easy setup of advanced security features in SFC. Virtual Network Functions (VNF) and SFC are used to implement these security features in BEACON. The chaining of federated cloud network security services is described in the OpenStack based service manifest. In the future we will try to translate this manifest in TOSCA.

In BEACON Project we are customising network security capabilities using network virtualisation technologies such as VXLAN and GENEVE in conjunction with Network Function Virtualisation (NFV) and Service Function Chaining (SFC). By combining NFV/SFC with network virtualisation it is possible for any cloud tenant to tailor the security of each of their individual virtual networks. In our model, aimed at OpenStack context, a tenant specifies his/her requirements through the Heat Orchestration Template (HOT), which is modified accordingly to our security functionalities. The IETF SFC working group [1] has defined a framework for SFC operation and administration. The Open Network Foundation has proposed SFC architecture. Their L4-L7 Service Function Chaining Architectural document defines the way of steering network traffic among several elements in order to get the desired service. In their example any end-to-end application traffic flows are required to traverse various network services such as IPS/IDS, firewalls, WAN optimizers and load balancers using the SFC. A user can make a decision in which the flow accessing an application server at a remote site may need to go through WAN optimizers for performance, firewalls and an IPS for security, load balancers for performance and high availability, and a VPN server before reaching the application server. The proposed general architecture

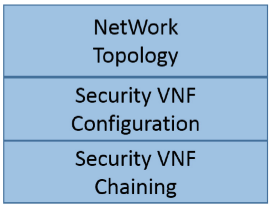
A. Lazovik and S. Schulte (Eds.): ESOCC 2016 Workshops, CCIS 707, pp. 305–307, 2018.
https://doi.org/10.1007/978-3-319-72125-5



(a) L4-L7 Service Function Chaining Architecture: details of the SF Instance Manager [1].

(b) BEACON Project solution [2].

Fig. 1. SFC, the IETF working group solution and BEACON Projects application scenario.



(a) HEAT Manifest Template (HOT) with Security features.

```
private void extractResourcefromManifest(JSONObject res)throws JSONException{
    String str = null;
    Iterator it_res=res.keys();
    JSONObject tmp=null;
    //new org.json.JSONObject(res.toString());
    while(it_res.hasNext()){
        String key=(String)it_res.next();
        tmp=res.getJSONObject(key);
        str=tmp.getString("type");
        switch(str){
            case "OS::Beacon::ServiceGroupManagement":{
                this.table_resourceset.get("OS::Beacon::ServiceGroupManagement").put(key, tmp);
                break;
            }
            case "OS::Beacon::fedNetManagement":{//BEACON>>> Manage it after Haifa meeting
                this.table_resourceset.get("OS::Beacon::fedNetManagement").put(key, tmp);
                break;
            }
            case "OS::Beacon::PoliciesAccManagement":{//Not used in this moment
                this.table_resourceset.get("OS::Beacon::PoliciesAccManagement").put(key, tmp);
                break;
            }
            case "OS::Beacon::fedSecManagement":{//Not used in this moment
                this.table_resourceset.get("OS::Beacon::fedSecManagement").put(key, tmp);
                break;
```

(b) A view of OS::BEACON Group template in HOT

Fig. 2. The HOT Manifest and HOT Parser in BEACON Project: resources, federation and security management sections.

can be applied to any network. However, in order to implement this architecture in federated heterogeneous networks, there is a need to coordinate SFC network controllers and classifiers, so they will speak the same language and deploy and control NFVs in an efficient way.

In our approach we are considering to use HOT of OpenStack for configuring SFC in the security domain. At our knowledge no any formalization has been done in this perspective yet. Figure 1(a) shows how our assessment is focused respect to the L4-L7 Service Function Chaining Architectural Scenario (see the rectangular shape over SF Instance Manager). Whereas Fig. 1(b) shows a typical scenario where BEACON Project operates.

Figure 2(a) shows the part we included in the HEAT Manifest Template. Starting with the conventional one (HOT) we extended it with Security VFN configuration along with Security VNF Chaining. Figure 2(b) depicts a Java code dealing with the new groups added for BEACON (see in the picture the OS:BEACON items). The original HOT is in YAML, but for our development we translated it into JSON format to be easily stored and retrieved exploiting

MongoDB. MongoDB represents the BEACON DB where we store the SFC-SF Instance Manager configurations.

In the near future BEACON Project is looking at the possibility to translate the HOT service definition template of OpenStack towards TOSCA, the Oasis Standardization initiative applied in this domain. Samuel Bercovici and Nati Shalom [3] ha an interesting assessment and experimentation in their work presented during an OpenStack summit on how to orchestrate Application and Network using HEAT and TOSCA. Our plan is to start from there.

Acknowledgment. This work has been supported by the BEACON project, grant agreement number 644048, funded by the European Union's Horizon 2020 Programme under topic ICT-07-2014.

References

1. Aldrin, S., Krishnan, R., Pignataro, N.A.C., Ghanwani, A.: Service function chaining operation, administration and maintenance framework. In: IETF RFC (2016)
2. Celesti, A., Levin, A., Massonet, P., Schour, L., Villari, M.: Federated networking services in multiple openstack clouds. In: Celesti, A., Leitner, P. (eds.) ESOCC 2015. CCCIS, vol. 567, pp. 338–352. Springer, Cham (2016)
3. Bercovici, S., Shalom, N.: When networks meet apps. http://docslide.us/technology/when-networks-meet-apps-samuel-bercovici-nati-shalom.html

Author Index