Grant Agreement N° 215483

Title:	*Validated SBA engineering principles exploiting HCI and contextual knowledge*
Authors:	*Tilburg, CITY, FBK, Lero, POLIMI, TUW, USTUTT, VUA*
Editor:	*A. Kounkou and N. Maiden (CITY)*
Reviewers:	*Fabrizio Silvestri (CNR)*
	Dominik Meil (Munster)
Identifier:	*Deliverable # CD-JRA-1.1.8*
Type:	*Deliverable*
Version:	*1.0*
Date:	*29/02/2012*
Status:	*Final*
Class:	*External*

Management Summary

In this deliverable, we present research performed in the last year on SBA engineering principles exploiting human-computer interaction and contextual knowledge; the work reported here builds upon and consolidates previous work based on validation results. Some of the presented contributions use and validate the S-Cube lifecycle model in the context of cloud computing. Other contributions focus on service-based application adaptation and evolution through context modelling or a change management methodology. Further contributions focus on practices and challenges of service-oriented architecture migration in industry and contrast it with academic practices. Finally, some research contributions investigate challenges in global software development and the related opportunities and relationship that exist with service-oriented architectures.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 215483 (S-Cube).

Members of the S-CUBE consortium:

University of Duisburg-Essen (Coordinator)	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
Vrije Universiteit Amsterdam	Netherlands

Published S-CUBE documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL: http://www.s-cube-network.eu/results/deliverables/

The S-CUBE Deliverable Series

Vision and Objectives of S-CUBE

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.
- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with today's limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-CUBE materials are available from URL: http://www.s-cube-network.eu/

Table of Contents

Table of illustrations

List of acronyms

A&M	Adaptation and Monitoring
ASN	Agile Service Network
BPEL	Business Process Execution Language
BPM	Business Process Management
CEP	Complex Event Processing
EAI	Enterprise Application Integration
GSE	Global Software Engineering
GUI	Graphical User Interface
KPI	Key Performance Indicator
PPM	Process Performance Metric
QA	Quality Assurance
QoS	Quality of Service
ROIA	Real-Time Online Interactive Applications
SC	Service Composition
SC&C	Service Composition and Coordination
SI	Service Infrastructure
SLA	Service Level Agreement
SN	Service Network
SOA	Service Oriented Architecture
SOSE	Service-Oriented System Engineering
TSE	Traditional Software Engineering

1 Introduction

1.1 Overview of the deliverable

This deliverable reports and summarizes the published joint research publications from JRA-1.1. More specifically it reports ten papers produced from partner collaborations in JRA1.1 over the last reporting period. In contrast to the larger numbers of papers reported from this activity in previous periods that described technical research advances, the papers reported in this deliverable are on broader themes that will enable the effective exploitation and uptake of the earlier research. Four of the ten papers report research into the challenges that enterprises face to migrate to service-oriented technologies and the challenges posed by global software development for service-oriented computing in this expansion. The remaining six provide principles and foundations for service-oriented computing, and an application of the S-Cube life-cycle model to cloud computing applied to real-time interactive applications. The rest of this section reports these new research outcomes in more detail, and where relevant links them to the JRA1.1 research challenges associated initially reported in deliverable CD-IA3.1.3.

The deliverable contains three research papers that focus on the relationship between global software development and service-oriented architectures. In the first, entitled **Using the cloud to facilitate GSD challenges,** we report new challenges that global software development poses. With the expansion of national markets beyond geographical limits, success of any business often depends on using software for competitive advantage. Furthermore, as technological boundaries are expanding, projects distributed across different geographical locations have become a norm for the software solution providers. Nevertheless, when implementing Global Software Development (GSD), organizations continue to face challenges in adhering to the development life cycle. The advent of the internet has supported GSD by bringing new concepts and opportunities resulting in benefits such as scalability, flexibility, independence, reduced cost, resource pools, and usage tracking. It has also caused the emergence of new challenges in the way software is being delivered to stakeholders. Application software and data on the cloud is accessed through services which follow SOA (Service Oriented Architecture) principles. In this paper, we present the challenges encountered in globally dispersed software projects. Based on goals mutually shared between GSD and the cloud computing paradigm, we propose to exploit cloud computing characteristics and privileges both as a product and as a process to improve GSD.

In the second paper, entitled **Going global with agile service networks (ASNs),** we report that ASNs are emergent networks of service-based applications (nodes) which collaborate through agile (i.e. adaptable) transactions. Global software engineering (GSE) comprises the management of project teams distanced in both space and time, collaborating in the same development effort. The GSE condition poses challenges that are both technical (e.g. geo-localization of resources, information continuity between time zones, etc.) and social (e.g. collaboration between different cultures, fear of competition, etc.). ASNs can be used to support global software engineering and build an adaptable social network (ASN_{GSE}) supporting the collaborations (edges of ASN_{GSE}) of GSE teams (nodes of ASN_{GSE}). Agile Service Networks can be used to support Global Software Engineering (GSE). This work contributes to overcoming the research challenge to support agile service networks with context modeling. In the third paper, entitled **Global software engineering: coordinating organizations or skills?**, we report on organisational challenges related to GSE. We mapped 25 GSE organizational challenges on results from a systematic literature review of organizational social structures, and found that the GSE condition creates a social structure in which project teams – which are distanced in both space and time - are aggregated into a network of practice shaped as a knowledge community of formal groups. Through this mapping a series of social structure requirements are extracted, and we found that new requirements concern skills' retrieval, visibility and shaping. This trend indicates that governance focus in GSE should be shifted towards skills rather than organizations. We also found that some organizational challenges are left unmatched. This indicates that further research should be invested in constructing an ad-hoc social structure, hybrid of current organizational social structure types, to match all organizational challenges.

Two related papers report challenges facing the uptake of and migration to service-oriented computing in industry. Both contribute new S-Cube knowledge to overcome the established research challenge identification of best practices for SOA migration. In the first, entitled **The how and why of SOA migration in industry,** we report that the migration of legacy software to service-based systems is an increasingly important problem area. So far, many SOA migration approaches have been proposed in both industry and academia. There are, however, considerable differences between SOA migration approaches defined in academia and those emerged in industry. This difference pinpoints a potential gap between theory and practice. To bridge this gap, we conducted an industrial interview survey in seven leading SOA solution provider companies. Results have been analyzed with respect to migration activities, the available knowledge assets and the migration process. In addition, industrial approaches have been contrasted with academic ones, hence discussing differences and promising directions for industry-relevant research. As a result we found that, in fact, all companies converge to one common SOA migration approach. This suggests that, with experience, enterprises mature toward a similar approach to SOA migration. The second paper is entitled **A survey of SOA migration in industry**. In industry, enterprises have many software systems to be modernized and made available as added-value services. The identification of migration strategies and practices for service engineering is critical for successful legacy migration, and SOA adoption in industrial setting. This paper presents the results of an interview survey on the migration strategies in industry. The purpose of this paper is two-fold: 1) to discover the migration strategies that industrial practice adopts, and: 2) to identify the uses of making such strategies explicit. Results of the survey have been analyzed with respect to migration activities, the available knowledge assets and the migration process. As a result we found that, in fact, all companies converge to the same, one, common SOA migration strategy. In addition, the uses of the strategy pinpoint promising industry- relevant research directions.

In the paper entitled **Using a lifecycle model for developing and executing real-time online applications on clouds,** we describe how the generic lifecycle model developed in the S-Cube project for the design and management of service-based applications (SBA) can be utilized in the context of cloud computing and Real-Time Online Interactive Applications (ROIA). In particular, we focus on the fact that the Infrastructure-as-a-Service approach enables the development of ROIA, which include multi-player online computer games, interactive e-learning and training applications and high-performance simulations in virtual environments. We illustrate how the lifecycle model expresses the major design and execution aspects of ROIA on clouds by addressing the specific characteristics of ROIA: a large number of concurrent users connected to a single application instance, enforcement of Quality of Service (QoS) parameters, adaptivity to changing loads, and frequent real-time interactions between users and services. We describe how our novel resource management system RTF-RMS implements concrete mechanisms that support the developer in designing adaptable ROIA on clouds according to the different phases of the lifecycle model. Our experimental results demonstrate the influence of the proposed adaptation mechanisms on the application performance. The paper directly contributes new S-Cube knowledge to overcome the challenge definition of a coherent life cycle for adaptable and evolvable SBA and measuring, controlling, evaluating and improving the life cycle and the related processes.

The final four research papers address services and service-based applications adaptation. In the paper entitled **Managing evolving services**, we motivate the need for a methodology to manage changes and variations so that impacted services in a service chain are appropriately (re-)configured, aligned and controlled. We outline sources and impact of change for services, review the concept of evolution in software and services, and hone in on adaptation as a mechanism for addressing service evolution. We report that due to services' strongly encapsulated and loosely coupled nature, compatibility and versioning become important mechanisms for enabling the seamless update of a service without affecting its existing consumers. However, since such changes are not always possible, we introduce a *change-oriented service lifecycle* capable of handling functional (structural and behavioural), non-functional, policy-induced and operational changes in order to support service developers to consider the scope and impact of changes and weigh their outcome against the effort and resources required to apply them. In the paper entitled **Addressing highly dynamic changes in service-oriented systems: towards agile evolution and adaptation,** we set out to introduce relevant foundations concerning evolution and adaptation of service-oriented systems. The paper starts by sketching the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service-oriented systems. Then, it provides an overview and more thorough explanation of the various kinds of changes that may need to be faced by service-oriented systems. To understand how such changes could be addressed, the chapter introduces a reference service life-cycle model which distinguishes between evolution, i.e. the manual modification of the specification and implementation of the system during design-time, and (self-)adaptation, i.e. the autonomous modification of a service-oriented system during operation. Based on the discussion of the key activities prescribed by that life-cycle, the chapter elaborates on the need for agility in both adaptation and evolution of service-oriented systems. Finally, in the paper entitled **A variable context model for adaptable service-based applications**, we present an adaptive approach to context modelling for adaptable SBAs. Context can be defined as characteristic information that is relevant to the interaction between a user and an application; it is an important factor for the selection of services and the execution of SBAs that can adapt to changes in a user's context and, consequently, changes in their requirements for the application. A context model enables the identification of context information to be collected and monitored, but the relevant contextual information may itself vary with the context. To address this, we propose granular analyses of context data and an approach to context modelling that is adaptive depending on the current situation. In the paper entitled **Exploiting codified user task knowledge to discover services**, we describe at length a mechanism for adapting service discovery to different user tasks, then report a first evaluation of the mechanism using precision and recall measures. The paper is a substantial journal extension to a paper first described as part of the year-3 deliverables. The results revealed that the service discovery algorithm extended with user tasks was more effective than the original algorithm under certain conditions. Whilst extending service discovery with user task models improved performance over keyword-based algorithms, it did not increase performance over an algorithm using sophisticated word sense disambiguation and term expansion algorithms from information retrieval. This result has implications for the relative cost-effectiveness of using user task models in service-based application development over other, potentially cheaper approaches.

The remainder of the deliverable is structured as follows. Section 1.2 describes the relationships between the research presented in this deliverable and other S-Cube work packages. Section 2 describes the research work carried out; table 1 (below) shows the correspondence between its subsections, corresponding research papers, and their topics. Section 3 concludes the report and relates the presented contributions to the challenges defined for the work package.

Main topic	**Paper title**	**Section**
S-Cube life cycle	• Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds	2.1
Adaptation and Evolution	• Addressing highly dynamic changes in service-oriented systems: Towards agile evolution and adaptation • Managing evolving services • A Variable Context Model for Adaptable Service-Based Applications • Exploiting Codified User Task Knowledge to Discover Services	2.2
Global software development	• Using the Cloud to Facilitate Global Software Development Challenges • Going Global with Agile Service Networks • Global software engineering: coordinating organizations or skills?	2.3
SOA migration	• A Survey of SOA Migration in Industry	2.4

	• The How and Why of SOA Migration in Industry	

Table 1. Research contributions presented in this deliverable

1.2 Relationships with other work packages

The contributions summarized in this deliverable can be related to research carried out in other S-Cube work packages as follows:

- The research presented in Section 2.1 relates to IA 3.1 which defines the reference life-cycle; it further contributes to IA 3.2 which compiles usage scenarios of S-Cube results, and validates those results, as well as to JRA 1.2 for the adaptation mechanisms presented.
- The research presented in Section 2.2 contributes to JRA 1.2 for the adaptation principles and mechanisms presented, and to JRA 1.1 for the application of agile methods for the development of adaptable SBAs.
- The research presented in Section 2.3 contributes to JRA 1.2 as it proposes agile context awareness strategies for adaptation, and JRA 2.1 as it focuses on supporting business processes.
- The research presented in Section 2.4 contribute to IA 2.2 as it focuses on European industry practices.

2 Research contributions

2.1 Exploiting the S-Cube life-cycle

Using a lifecycle model for developing and executing real-time online applications on clouds

Service-oriented applications are developed for constantly changing environments with the expectation that they will evolve over time. Several service-oriented system engineering (SOSE) methodologies have been proposed aiming at providing methods and (sometimes) tools for researchers and practitioners to engineer service-oriented systems. SOSE methodologies are more complex than traditional software engineering (TSE) methodologies: the additional complexity results mainly from open world assumptions, co-existence of many stakeholders with conflicting requirements and the demand for adaptable systems. A number of service lifecycle models have been proposed by both industry and academia. However, none of the proposed models has either reached a sufficient level of maturity or been able to fully express the aspects peculiar to SOSE. Within the S-Cube project a new Lifecycle Model was designed that combines existing techniques and methodologies from TSE and SOSE to improve the process through which service-based applications will be developed.

This paper extends our previous work on studying how the S-Cube Lifecycle Model can be applied on the emerging and challenging domain of *Real-Time Online Interactive Applications (ROIA)* including multi-player online games, high-performance simulations, e-learning applications, etc. In particular, we study how to use server resources economically, which is difficult due to continuously changing user numbers.

Cloud Computing with its *Infrastructure-as-a-Service (IaaS)* approach offers new opportunities for ROIA execution and promises a potentially unlimited scalability by distributing application processing on an arbitrary number of resources given suitable adaptation mechanisms. Clouds allow for adding/removing resources on demand. This opens for ROIA an opportunity to serve very high numbers of users and still comply with QoS demands. Despite a variable number of users, Cloud resources can be used efficiently if the application supports adding/removing resources during runtime. Hence, using Cloud Computing for resource provision and the Lifecycle model for implementing adaptable ROIA complement each other.

This paper studies how Cloud Computing and the S-Cube Lifecycle Model can be utilized for ROIA applications. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds and present our novel resource management system RTF-RMS that implements concrete mechanisms for ROIA development and adaptation according to the Lifecycle. We report experimental results on the influence of the proposed adaptation mechanisms on the application performance.

2.2 Supporting SBA evolution and adaptation

Addressing highly dynamic changes in service-oriented systems: towards agile evolution and adaptation

Modern software technology has enabled us to build software systems with a high degree of flexibility. The most important development in this direction is the concept of service and the Service-oriented Architecture (SOA) paradigm. A service-oriented system is built by “composing” software services (and is thus also called “service composition” or “composed service” in the literature). Software services achieve the aforementioned high degree of flexibility by separating ownership, maintenance and operation from the use of the software. Service users do not need to acquire, deploy and run software, because they can access its functionality from remote through service interfaces. Ownership, maintenance and operation of the software remains with the service provider. While service-orientation offers huge benefits in terms of flexibility, service-oriented systems face yet another level of change and dynamism. Services might disappear or change without the user of the service having control over such a change. Agility, i.e., the ability to quickly and effectively respond to changes, will thus play an ever increasing role for future software systems to live in the highly dynamic “world” as sketched above. Agility can be considered from two view- points:

- First, agility may concern the evolution of the system. This means that it concerns the development process and how engineering activities (such as requirements engineering and implementation) should be performed to timely address changes by evolving the software;
- Secondly, agility may concern the adaptation of the system. This means that it concerns the system itself and how the system should respond to changes. Agility in adaptation is typically achieved through self- adaptation, i.e., the autonomous modification of a service- oriented system during operation.

This work sets out to introduce relevant foundations concerning evolution and adaptation of service-oriented systems. It starts by sketching the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service-oriented systems. Then, it provides an overview and more thorough explanation of the various kinds of changes that may need to be faced by service-oriented systems. To understand how such changes could be addressed, the chapter introduces a reference service life-cycle model which distinguishes between evolution, viz. the manual modification of the specification and implementation of the system during design-time, and (self-)adaptation, viz. the autonomous modification of a service- oriented system during operation. Based on the discussion of the key activities prescribed by that life-cycle, the chapter elaborates on the need for agility in both adaptation and evolution of service-oriented systems.

Managing evolving services

Services are subject to constant change and variation, leading to a continuous service re-design and improvement effort. Service changes originate from different sources such as introducing new functionalities to an existing service, modifying the current functionality of a service in order to improve its performance, or including new regulatory constraints requiring a change of the behavior of services. Such changes lead to a continuous service re-design and improvement effort. However, they should not be disruptive by requiring radical modifications in the very fabric of services, or the way that business is conducted between service providers and consumers.

With the term service evolution we refer to the continuous process of development of a service through a series of consistent and unambiguous changes. The evolution of services is expressed through the creation and decommissioning of service versions during their lifetime. These versions have to be aligned with each other in a non-disruptive manner and in a way that allows a service designer to track the various modifications and their effects on the service in terms of consistency. Looking at the effect that service changes have on their consumers in the Service Oriented Architecture (SOA) paradigm, we can classify them as shallow and deep. Shallow changes are incremental changes, localized to one service and restricted to the consumers of the service. Such incremental changes are based on the notion of compatibility between service versions to allow for a seamless and transparent to the service clients update of the service. Shallow changes therefore require a robust versioning scheme and an unambiguous definition of compatibility in order to be managed efficiently. Deep changes, on the other hand, are transformational changes, cascading beyond the clients of the service and potentially to entire end-to-end service networks. As such, they require a change-oriented service development methodology that considers the scope, effect, effort and applicability of such changes in across enterprise environments and throughout entire service chains.

Despite its connections with component evolution, service evolution poses a number of additional challenges due to the strongly encapsulated and loosely coupled systems (i.e. services) that it deals with. In this context, service compatibility and versioning become important mechanisms for enabling the seamless update of a service without affecting its existing consumers. Such changes are not always possible however. In this case it is required of service developers to consider the scope and impact of the change and weigh the outcome against the effort and resources required for applying it. A systematic change-oriented service lifecycle should be used for this purpose. Describing the key concepts in dealing with evolving services, we provide a sound foundation for a change-oriented service lifecycle to spread changes in an orderly fashion so that impacted services in a service-chain are appropriately (re-)configured, aligned and controlled as the changes occur.

A variable context model for adaptable service-based applications

Context is defined as: "any information that can be used to characterize persons, places or objects that are considered relevant to the interaction between a user and an application, including users and applications themselves". Context should be taken into account for service selection. For example, the service that informs a sailorman about the weather forecasts on a specific route should be very detailed and focused on the conditions of the sea and of winds, while the service dedicated to a family willing to decide if to book a trip on the seaside should be focusing on the weather conditions of a specific place, typically, on a longer time scale.

More generally, service-based applications should be able to adapt the execution flow to address changes of the execution context. For example, applications have to be flexible in order to satisfy users' variable requirements on the basis of the situation (e.g., geographical position, time) in which users are when they access the application.

A first goal of this paper is to provide a novel context model for adaptable service-based applications and to point out to its role in the adaptation activities. The context model is the basis for the definition of triggers enabling adaptation or evolution of service-based applications and enables the identification of the information that has to be collected and monitored at run-time. We propose an approach to context modeling which is itself adaptive to the current situation. In fact, the relevant contextual information might be different in different situations: for instance, the information needed about a location may be different depending on the location (e.g., a small village vs a large town), or depending on the user who is interacting with the application (e.g., a user who knows a location well vs a user who is not familiar at all with the location). In these cases the way the application behaves might vary not only according to the context in general, but according to the representation of the context itself that is variable in the different situations.

Context data are gathered by using different kinds of sensors. An important issue is to understand the level of granularity at which to collect data. The level of granularity is defined by the amount of details to catch for representing the significant characteristics of the environment for a given application. In fact, it is convenient to avoid the collection of unnecessary details of data that are not suitable to catch changes in the execution context. For example, in case of emergency situations, it is necessary to collect data values every second in order to have a fine control of the situation, while in other situations a daily value is enough since a more frequent measure of context data does not provide any additional relevant information. In location-based applications, in some cases, coarse information about the location (e.g. a country) may be sufficient, while in other cases it is important also to be aware of the regional context.

As a result of the previous considerations, another goal of this paper is to address the issue of the granularity of monitored context data and to propose a way to support adaptation, by analyzing context data at different granularities and using a proactive approach to establish the adaptation needs and the dynamic invocation of the services.

Exploiting codified user task knowledge to discover services

This research developed a new software-based algorithm for adapting service discovery to different user tasks, then reported a first evaluation of the mechanism using precision and recall measures. The paper is a substantial journal extension to a paper first described as part of the year-3 deliverables. The results revealed that the service discovery algorithm extended with user tasks was more effective than the original algorithm under certain conditions. Whilst extending service discovery with user task models improved performance over keyword-based algorithms, it did not increase performance over an algorithm using sophisticated word sense disambiguation and term expansion algorithms from information retrieval. This result has implications for the relative cost-effectiveness of using user task models in service-based application development over other, potentially cheaper approaches.

In detail we report the development and codification of user task models developing using the Concurrent Task Trees (CTT) approach and their application to service discovery in one environment developed to design service-based applications. The user task models were developed at the class-level (e.g. drive to a destination) to maximize the leverage of each model during service discovery – one model could potentially be exploited during the design of all service-based applications that instantiate that task class (*e.g. drive from London to Paris via the Channel Tunnel*). The codified user task models were documented in a searchable catalogue, then service queries were generated and fired at a service repository. An empirical evaluation explored the effect of modifying service queries with codified user task models on the precision and recall of one service discovery engine.

The algorithm for service discovery based on user task models was trialed against an existing service discovery algorithm that could be configured to different settings. Evaluation data was to accept or reject four research hypotheses that informed the experiment. Results revealed that extending rather than replacing service queries with additional knowledge about user tasks did improve the overall effectiveness of service discovery. However, the reformulation of service queries with knowledge about user tasks did not decrease the number of irrelevant services retrieved by the service discovery engine. We partially accepted the hypothesis that the reformulation of service queries with knowledge about user tasks would increase the number of relevant services retrieved by a service discovery engine in some but not all conditions. We also partially accepted the hypothesis that the reformulation of service queries with knowledge about user tasks would improve the overall correctness of services retrieved by a service discovery engine, again in some but all conditions. These complex results reveal that context knowledge about the user task expressed as class-level user task models improved service discovery, but no more than current other sophisticated service discovery algorithms do. They raise implications about the overall cost-effectiveness of the role of user task models in service-based application development and use that are explored in the paper.

2.3 Supporting global software development

Using the cloud to facilitate global software development challenges

The expansion of software and service markets beyond geographical limits has given rise to use of software for competitive advantage. Furthermore, expanding technological boundaries have changed the way software and business solutions are developed; the advent of internet for instance has brought in new methodologies that result in business advantages and reduced costs, but organizations very often face difficulties due to global distance and the involvement of development teams which are geographically distributed. Advances in technology and communication channels have made business organizations to outsource software development operations in multiple geographical locations as the exchange of information has become more accurate and available. However, outsourcing development operations to organizations at various outsourcing destinations is not an easy and straightforward task. While implementing Global Software Development (GSD), software organizations continue to face challenges in adhering to the development life cycle. It has also caused the emergence of new challenges in the way software is being developed and delivered to the stakeholders. GSD is software development incorporating teams spread across the globe in different locations, countries, and even continents. The business models in low cost countries have provided with capable and cheap work force to reap the benefits of outsourced and offshore software development. With the emergence of technologies in a world which has become increasingly globalized, the relationship between culture and management of remote work has become an unavoidable issue which needs to be addressed. Because of distance among the software development teams, GSD encounters certain challenges in terms of collaboration, communication, coordination, culture, management, organizational, outsourcing, development process, development teams, and tools.

In order to conduct this research, our literature review studied characteristics of services (both SOA and the cloud). We also identified challenges faced by GSD and held a workshop, attended by all of the authors of this paper, each of whom has research and/or industrial expertise in GSD and/or SOA. During this workshop, through interactive discussion and brainstorming, we developed the concepts presented in this paper. To do this, we summarized the GSD challenges and requirements and investigated the potential of SOA based cloud services to address these.

Collaboration Challenges	Issues	Negative Impact on Software Project	Facilitating GSD Using Services (SOA/Cloud)
Geographic	Distance Time Knowledge transfer Tools	Communication gaps Project Delays Ambiguity on technical aspects Unequal quality levels across the software development sites	Dynamic binding, runtime adaptation, and timely availability of required services could help dealing with geographic issues. Also, availability of SaaS could diminish installation overheads at each development location.
Cultural	Unequal distribution of work Lack of Trust, Fear	Increase in cost Poor skill management Reporting problems	Service could maintain a fair distribution of work between the teams. Only a specific person will be responsible for the task assigned to thus skill management would be easier too.
Linguistics	Frequency of communication Knowledge transfer	Loss in project quality Invisibility on project development Ineffective project management	Run time evolution of services can meet with the linguistic issues. Also, isolation of each task and related information as a service can ensure right level of knowledge transfer.
Temporal	Lack of Motivation Less visibility	Loss in project quality Poor management of configuration Chances of	The cloud service models imply that the data resides on a centralized location where inventory of services is maintained. Services

	Risk	project artifact loss	maintain a registry where all of them are stored. This attribute could be used to store and retrieve configurations.

Table 2. GSD challenges possibly facilitated by the use of services

We identified different challenges associated with GSD and suggested the likelihood of using the cloud paradigm to address them. Different GSD development activities were figured out; since information and data on the cloud is transmitted and shared by means of web services which work on underlying SOA principle, we take advantage of its benefits like loose coupling, service composition, and negotiation to facilitate software development practices across multiple development sites in light of different cloud service models that include IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) and its characteristics like scalability, performance, virtualization, and reduced costs .

Virtualization	Courtesy of this privilege, cloud providers can enhance their infrastructure to accommodate in case there is growing demand for services. Usually, a combination of hardware and software are used on the provider side to meet with the scaling requirements.
Reduced Cost	Costs in the cloud do not include server side infrastructure and equipment costs. Moreover, pay as you go model ensures that subscribers are bound to pay for only those resources which they use. In short, the distribution costs of software are reduced.
Scalability	On-demand provision of application software provides scalability, which results in greater efficiency. Whereas cloud based application development platforms provide with high level of scalability thus making the developed application to coup with the fluctuation demands.
Infrastructure	Providers' applications are run on a cloud infrastructure from where a consumer can access those. Similarly, consumer-modified information or application can be deployed on the same infrastructure as well. The privilege is that the consumer does not have to deal with the underlying infrastructure.
Performance	The cloud paradigm can support various levels of performance requirements like service scaling, response time, and availability of the application based on the needs of the consumers. In addition indirect performance measures may also be achieved by eliminating the overheads involved with installation procedures and reduction in unnecessary reduction among the applications running on the cloud
Multi Tenancy Support	Public clouds are elastic in nature as their consumers are not limited. More importantly, consumers' workloads are isolated to provide privacy. However, the number of consumers can be restricted by opting out a specific deployment model.

Table 3. Supporting characteristics of cloud computing for GSD

In addition, we consider an example scenario to understand the GSD collaboration challenges that could be minimized using the cloud paradigm. We suggest that using a cloud paradigm will result in GSD benefitting from the cloud's infrastructure, platform, and provision of software as a service features. We do not argue that the cloud paradigm can fully serve the purpose but we do believe that, if designed correctly, GSD can be successfully supported by services. Although the work to date have already laid some solid foundations, we are embarking on further research to understand whether these indeed can be of value to both the industrial and research communities.

Going global with agile service networks

A fundamental similarity can be identified between GSE and ASNs. Both stem from business decisions. Moreover, a crucial complementarity exists between them. On one side, GSE needs dynamism among nodes (development teams) and their collaboration towards business gain (timely delivery). On the other, ASNs are supporting dynamic collaborations among nodes which are teaming up to increase business gain.

Based on these considerations, we argue that GSE challenges can be overcome through an ASN-based social network (ASNGSE) providing agility of communications and collaborations (edges of ASNGSE) to globally located IT professionals (nodes of ASNGSE). Global professionals can be represented as nodes in an agile (i.e. adaptable and emergent) organizational social network to deliver the final product, just-in-time and sufficiently-good.

Distances in time and space make it impossible for GSE teams to communicate and coordinate their effort in an efficient manner. *How can ASNs be used to support GSE?*

The solution we advocate is an ASN-based social network (ASNGSE). This technology should exhibit four key characteristics:

1) *Agile context awareness*. ASNs are able to detect changes in the context and dynamically support different scenarios as needed. In GSE for instance, round-the-clock productivity could be supported by dynamically allocating collaborations between teams, by modeling each developer as a set of skills and allowing for their seamless (re-)allocation based on their timezone, location and needs. Also, seamless handoff of relevant informations between two contiguous timezones could be used to ease the coordination of sequential or dependent work packages.

2) *Deployed in the cloud*. Cloud computing, has potentials which fit with GSE needs. For instance, GSE resources rendered available in the cloud allow for rapid resource location and access on a global scale. Also, communication and information continuity between timezones may be requested as needed.

3) *Satisfying GSE social requirements*. GSE teams together create an Organizational Social Structure (OSS) [3], part of a global corporation. Social interactions in OSSs depend on personal- or corporate-specific practices, which include work habits, methods, technologies to support cooperation, etc. In GSE, for instance, supporting social interactions among developers from different companies and cultures, would require letting them use own tools, languages and own methods seamlessly. ASNs can help in doing that through adaptable creation of service compositions or transparent information proxying (i.e. providing seamless switching between answering nodes in case of difficulty). Another example could be using ASNs to autonomously try and assemble a common tool workbench for all teams. Lastly, an ASN to support GSD could compute the work allocation to teams, using (up-to-date and context-aware information) on project requirements, time constraints, current availability, etc.

4) *Project-centric as well as People-centric*. Enterprise Social Networking technologies already exist which could potentially represent (and supporting) the social network of an entire corporation. What is still missing is the dynamic / automatic adjustment of its granularity, to support the global software development project (against its changing context) as well as the people involved (e.g. technicians, developers, managers, etc.). Moreover, none of these technologies provide flexible and adaptable collaboration channels (e.g. adaptable status-tracking, always-on reachability of key roles, worldwide project chatter, etc.) among professionals collaborating in the same GSE effort.

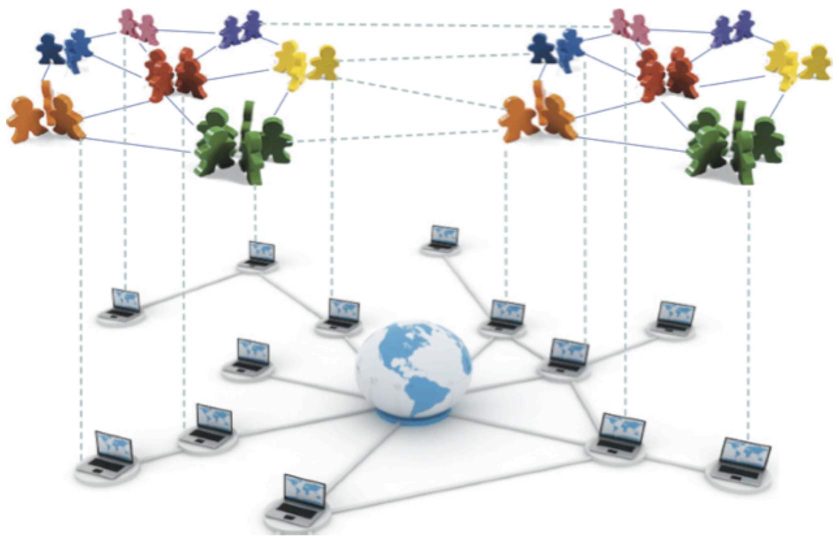


Figure 1. ASNs for GSD

We plan to develop a very initial version of a prototype for the proposed social-networking ASN for GSE, to initiate industrial action-research validation.

Global software engineering: coordinating organizations or skills?

Global Software Engineering (GSE) and service-oriented development are strongly related even though this relation has not been recognized yet in the literature.

GSE is a business decision which entails project teams to collaborate globally on the same project, in different timezones and continents. The need of this collaboration may come from various reasons ranging from the inherently decentralized organization of many multinational companies to the opportunity of exploiting some existing skills available in third countries, to the need of tailoring services to specific local regulations, to the possibility of incorporating existing delocalized services. The literature on GSE is focusing on addressing the many process and organizational challenges that arise in this setting and highlights the problems that are introduced by the spatial, temporal and cultural distances that inevitably occur within the global teams.

While the SBAs literature in most cases disregards the organizational problems and focuses mainly on technological issues, the development of a SBA where services are owned and operated by third parties is undoubtedly a GSE issue and, as such, it is appropriate to understand how GSE is declined in this specific case.

As part of the current work we propose an initial profile of the GSE Organizational Social Structure (OSS), using empirical data. We conducted a systematic literature review into OSS types and attributes; in addition, we used a set of 25 organizational challenges stemming from previous action research in the GSE field. Mapping GSE challenges on OSS types we found that the GSE social structure is the mix of four types identified in literature, namely: project teams, networks of practice, knowledge communities and formal groups (see Figure 2).

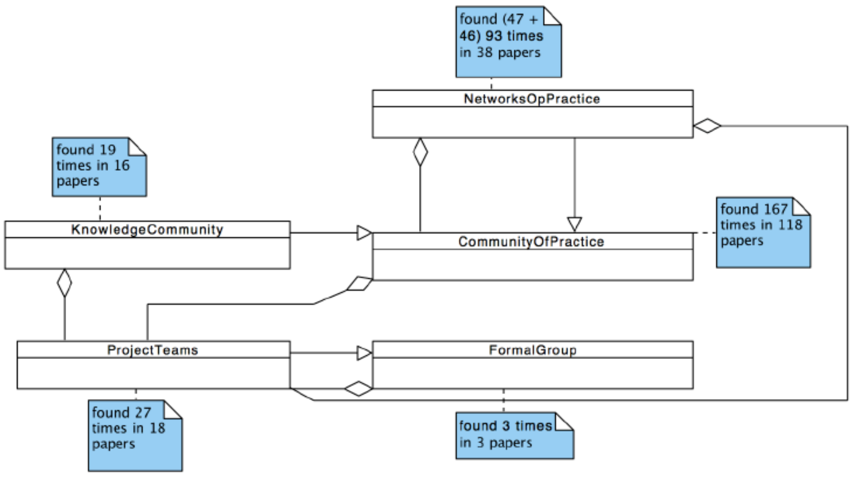

found (47 + 46) 93 times in 38 papers
NetworksOpPractice
found 19 times in 16 papers
KnowledgeCommunity
CommunityOfPractice
found 167 times in 118 papers
ProjectTeams
FormalGroup
found 27 times in 18 papers
found 3 times in 3 papers

Figure 2. OSS types: project team aggregates

By analyzing the characteristics of these OSSs and the known challenges for GSE we found out that while the current literature has deeply investigated the problem of coordination within the global team, the governance of skills in the global team is still an open research area that requires specific attention (see Table 4).

Factor	**Description**
Communication	the challenge here is in terms of communication. How open it should be? how should it be enforced or maintained?
Communication Tools	the challenge here is what tools shold be used. what communication paradigm should be considered and so on
Temporal Issues	the challenge here is how should round-the-clock productivity be maintained? how should a tool support this maintenance?
Effective Partitioning	how should work be split and spread across teams in different timezones / continents?
Skill Management	How should the work be spread in terms of skills?
Knowledge Transfer	how should knowledge sharing be nurtured?
Defined Roles / Responsibilities	how should roles and responsibilities be allocated to different engineers / skills across the project teams?
Team Selection	how should members in the teams be selected?
Motivation	how should motivation of GSD engineers be monitored and maintained?
Technical Support	what kind of technical support tools or specialized engineers should be deployed to maximize productivity?
Coordination	what kind of coordination issues might rise (for the specific project)?
Cooperation	what kinds of cooperation practices can be put in place (for the specific project)?
Culture	what kind of cultural practices should be considered / maintained?
Teamness	what team building practices should be put in place? how should teamness be maintained?
Visibility	how should visibility of the project be maintained? how should awareness be kept high?
Trust	what trust dynamics should be envisioned? what mechanisms should be put in place to maintain them?
Fear	what social fears might rise (for the specific project)? what kind of fear-fighting practices should be put in place?
Project Management	what tasks should be allotted to management? what should be allotted to local roles?
Effective Partitioning	how should work be split and spread across teams in different timezones / continents?

Risk Management	what risk management policies should be put into practice? what changes in the context might compromise GSD operativity?
Language	how should language difference be mitigated?
Selection tools	what tools should be made available to engineers? what should be the full technical space?
Information	what kind of information is to be created / shared / maintained?
True Cost	what is the estimated post-mortem cost of the project in any given moment of time?
Reporting Process	what kind of reporting should be procured for the GSD attempt? how should the software architecture be documented? how should the implementation be documented? what kind of process should be followed to maximize GSD effectiveness?

Table 4. 25 organisational factors in GSD

As part of the work we also explored current Enterprise Social Networking (ESNs) tools to see the extent to which they can be used to enable skills awareness within the global team.

2.4 SOA migration in practice

Service Oriented Architecture (SOA) migration from legacy systems to service-oriented software is not new. Many methods do exist, originated in both academia and industry. Companies have extensive experience in both *in-house migration* of their own legacy information systems to a more agile, reusable service-oriented paradigm, and *consultancy migration* to support customer organizations to port their systems to modern service-oriented technologies, make them available as added-value services, often with the goal of creating new market opportunities.

Many SOA migration approaches have been developed in both industry and academia. Nevertheless, we have observed that the industrial approaches are considerably different from the ones originated in academia. By discussing this observation with practitioners we were suggested that such differences might pinpoint an undesired gap between theory and practice. It is essential to fill this gap to devise solutions that fit the goals and problems of industry. This need was further emphasized most recently in a panel on ``What Industry Wants from Research'' in ICSE 2011. The general consensus among the panel members was that there is a need to better understand the fundamental problems, goals, strategies and weaknesses of practice.

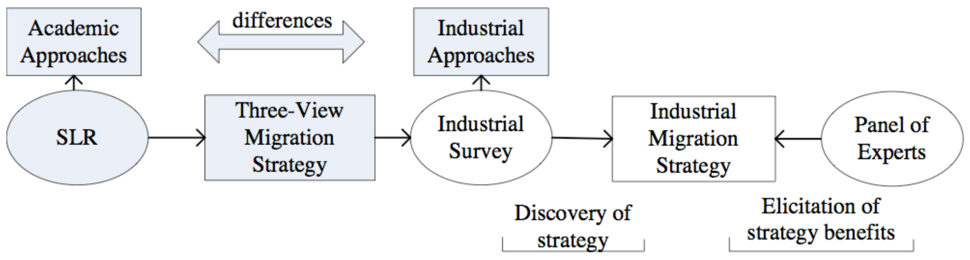


Figure 3. Research context

To gain an understanding on industrial migration strategies, we conducted an industrial interview survey in seven leading SOA solution provider companies. With the objective of understanding the industrial migration approaches, we designed and executed the interviews. As a result we found that despite the diversity of participating enterprises, they all converged to the same, one, common SOA migration strategy: all use similar input knowledge, similar activities, and sequences of activities to carry out migration. This suggests that with experience enterprises mature toward a similar migration approach. This would also confirm the SOA migration maturity level of Gartner Hype Cycle as being in early main stream phase. In addition, and unlike the majority of academic approaches, SOA migration in industry mostly neglects reverse engineering. Rather, migration follows a forward engineering process initiated by identifying the ideal state (e.g. ideal business services), which is taken as a reference to extract and transform legacy elements to services.

The panel of experts following the interview survey investigated the benefits of such overall strategy. The panel envisioned to use this strategy as a general tool to guide and steer migration projects. We further elicited a list of extensions to such tool, that would address the recurring problems in industrial migration, namely identification of the costs and risks of migration projects, and deciding on the best migration approach to mitigate them. The overall approach with extensions that emerges from the panel resembles the lean and mean approach of Kruchten, and draws interesting directions for industry-relevant research.

In addition, we contrasted the industrial approaches with academic ones, which we identified from a previous Systematic Literature Review (SLR) on SOA migration. Here we use the results of the SLR to discuss the differences and draw promising directions for industry-relevant research listed below.

*Migration approaches fitting activities carried out in industrial approaches***.** Migration activities that industrial approaches carry out can act as a frame of mind confining the migration approaches that are more aligned with practice. From that perspective, one would see that, for instance, the approaches addressing wrapping the applications as a whole are more in-line with practitioners concerns, compared to the ones addressing the automatic recovery of the legacy architecture. Hence, this frame of mind pinpoints the types of industry-relevant research in SOA migration methodologies and techniques.

To-Be driven migration approaches. Inadequate support for To-Be driven approaches in academia highlights promising opportunities for research to focus on how to support To-Be driven migration. For instance, future research can focus on addressing the following challenge of the practitioners: how to systematically elicit and capture the migration drivers and how to shape the migration process using those drivers.

Legacy understanding without reverse-engineering. Although reverse engineering is not covered in industrial migration approaches (see Figure 4), elicitation of the knowledge about the legacy system is crucial for a successful migration. In this regard, research can benefit practice by providing methods, techniques, or guidelines that facilitate elicitation of migration-relevant knowledge from different sources of such knowledge.

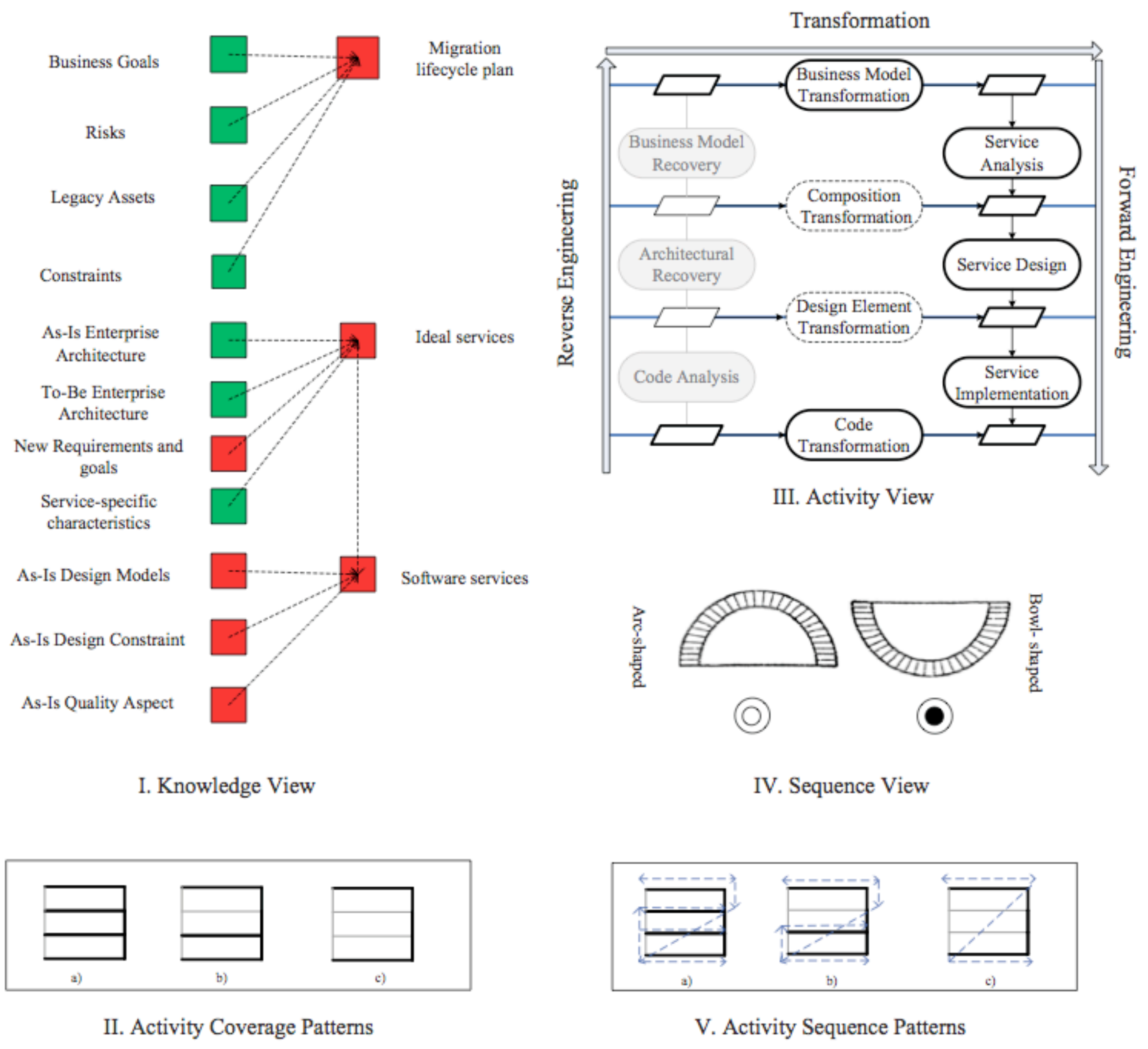


Figure 4. Three view Strategy Representation.

Legacy evaluation from multiple perspectives. Companies evaluate and extract the legacy assets for migration to SOA by depicting their ideal services. This is, however, done in an ad-hoc manner, which may hinder successful service extraction. An immediate concern calling for further research is how to systematically evaluate pre-existing legacy assets based on different aspects of the ideal services.

3 Conclusions

The research presented in this deliverable contributes further towards the ultimate fulfillment of JRA-1.1 challenges outlined in CD-IA-3.1.3 (First version of Integration Framework) and refined in CD-IA-3.1.5 (Consolidated Revised Integration Framework):

- *Definition of a coherent life cycle for adaptable and evolvable SBA and measuring, controlling, evaluating and improving the life cycle and the related processes*. The S-Cube lifecycle is utilized in Section 2.1 to express key aspects of the engineering of ROIA applications on clouds, and used as a foundation for the development of new mechanisms for their development and adaptation. The research presented in Section 2.3 further draws on it by its focus on software and services development life cycle phases, and on elements of the S-Cube research framework across JRA-1 and JRA-2 - Engineering and Design and Service Composition and Coordination. Finally, the research presented in Section 2.2. potentially extends the lifecycle into a change-oriented service lifecycle for the management of evolving services. As such we believe that these new research contributions at least in part plug previous gaps in our understanding of and prescriptive guidance for delivering a complete life-cycle for SBAs. Furthermore, given the layered approach used to generate these research contributions, we believe that the latest research is coherent with previous results because it builds on them;

- *HCI and context aspects in the development of service based applications*. The research presented in Section 2.2, more particularly **A variable context model for adaptable service-based applications**, focuses on enhancing the interaction between users and applications through an improved context modeling and analysis supporting SBA adaptation activities. At the end of S-Cube the substantial bodies of research in HCI has yet to have a significant impact on service-oriented computing research and development. The contributions made in S-Cube represent some of most substantive research in this direction. The attempt to develop a rigorous model of the large number of context variables represents a sizeable advance in this direction;
- *Identify best practices for SOA migration*. The work presented in Section 2.4 contribute to this research by addressing the gap between theory and practice of SOA migration through an industry survey and its analysis and contrast against academic practices. The resulting understanding of current challenges in SOA migration practices will, we believe, lead to important follow-on applied research and technology transfer activities amongst some of the S-Cube partners. Furthermore, the SOA migration challenges are now widely recognized as an impediment to research take-up in the field, and S-Cube has positioned itself as a valuable source of information, as well as outlet, for example the forthcoming ICSE'2012 tutorial on the subject;
- *Support agile service networks with context modeling*. The research presented in Section 2.3 on **Going global with agile service networks** directly addresses this challenge and lists agile context awareness as a key characteristic of their proposed ASN-based social network being prototyped. Clearly there is scope for more research post S-Cube in this direction, and again the traction and motivation generated by some S-Cube research partners will mean that more research to address context understanding that enables more agile future service networks will take place.

To conclude this deliverable builds upon and consolidates research presented in previous JRA 1.1 deliverables. Furthermore, it contributes to cross-package research integration as it relates to work carried out in other S-Cube work packages. We believe that this research offers the ideal springboard for future SOA research in important, valuable directions.

Appendices

Appendix A: Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds

D. Meiländer, A. Bucchiarone, C. Cappiello, E. Di Nitto, S. Gorlatch

1 Introduction

Service-oriented applications are developed for constantly changing environments with the expectation that they will evolve over time. Several service-oriented system engineering (SOSE) methodologies have been proposed aiming at providing methods and (sometimes) tools for researchers and practitioners to engineer service-oriented systems. SOSE methodologies are more complex than traditional software engineering (TSE) methodologies: the additional complexity results mainly from open world assumptions, co-existence of many stakeholders with conflicting requirements and the demand for adaptable systems. A number of service lifecycle models have been pro- posed by both industry and academia. However, none of the proposed models has either reached a sufficient level of maturity or been able to fully express the aspects peculiar to SOSE. Within the S-Cube project [1] a new Lifecycle Model was designed that combines existing techniques and methodologies from TSE and SOSE to improve the process through which service-based applications will be developed.

This paper extends our previous work [2] on studying how the S-Cube Lifecycle Model can be applied on the emerging and challenging domain of Real-Time Online Interactive Applications (ROIA) including multi-player online games, high-performance simulations, e-learning applications, etc. In particular, we study how to use server re- sources economically, which is difficult due to continuously changing user numbers.

Cloud Computing with its Infrastructure-as-a-Service (IaaS) approach offers new opportunities for ROIA execution and promises a potentially unlimited scalability by distributing application processing on an arbitrary number of resources given suitable adaptation mechanisms. Clouds allow for adding/removing resources on demand. This opens for ROIA an opportunity to serve very high numbers of users and still comply with QoS demands. Despite a variable number of users, Cloud resources can be used efficiently if the application supports adding/removing resources during runtime. Hence, using Cloud Computing for resource provision and the Lifecycle model for implementing adaptable ROIA complement each other.

This paper studies how Cloud Computing and the S-Cube Lifecycle Model can be utilized for ROIA applications. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds and present our novel resource management system RTF-RMS that implements concrete mechanisms for ROIA development and adaptation according to the Lifecycle. We report experimental results on the influence of the proposed adaptation mechanisms on the application performance.

The paper is structured as follows. We introduce the S-Cube Lifecycle Model in Section 2, followed by a description of the challenges in ROIA development and execution on Clouds in Section 3. Section 4 illustrates how the Lifecycle Model is applied for ROIA development on Clouds using RTF-RMS. Section 5 reports experimental results on the adaptation of a sample ROIA, and Section 6 concludes the paper.

2 Lifecycle Model

Many of the existing development methodologies for service-based applications (SBA) are based on the results carried out in the fields of classical software and system engineering and do not facilitate SBA adaptation [3-5]. Some of the reported SBA development approaches such as SOUP (Service

Oriented Unified Process) [6] or the approach by Linner et al [7] do support some level of adaptation, however, they lack sufficient process details. Lane and Richardson [8] carried out a systematic literature review of SBA development approaches, they identified 57 such approaches of which there were only eight that specifically dealt with adaptation. Only four of these eight approaches target the adaptation of SBAs, the others target the adaptation of services.

Each of the four approaches shows interesting features, but even those that enable the definition of various adaptation strategies lack a coherent approach to support de- signers in this complex task. Moreover, they focus on the implementation process with- out considering what impact adaptation has on the rest of the development and operational lifecycle [9-11]. Finally, they also tend to focus on particular types of adaptation, such as adaptation due to requirement violations [12], or service substitution due to application constraints [13], so it is difficult to elicit generic adaptation mechanisms from them. In summary, each of these approaches focused on the analysis and design pro-cesses without consideration for any other development or runtime processes.

The Lifecycle Model proposed in the S-Cube project (see Fig. 1) aims to support the design of adaptable SBAs [14]. It provides a solid reference [15] for practitioners who are planning to develop adaptable SBAs since that it has advantages over similar approaches in that it focuses on software process rather than the specific adaptation mechanism implementation techniques. It highlights not only the typical design-time iteration cycle, but it also introduces a new iteration cycle that is performed at runtime when the application needs to be adapted on-the-fly. Both cycles coexist and support each other during the lifetime of the application. In particular, the design time activities allow for evolution of the application, i.e., introduction of permanent and, usually, important changes, while the runtime activities allow for temporary adaptation of the application to a specific situation. At design time, it is important to analyze functional and non-functional requirements, context and machine characteristics in order to (i) identify the types of changes that trigger self-adaptation, (ii) define mechanisms to monitor the environment and the system behavior, and (iii) develop strategies for self-adaptation.

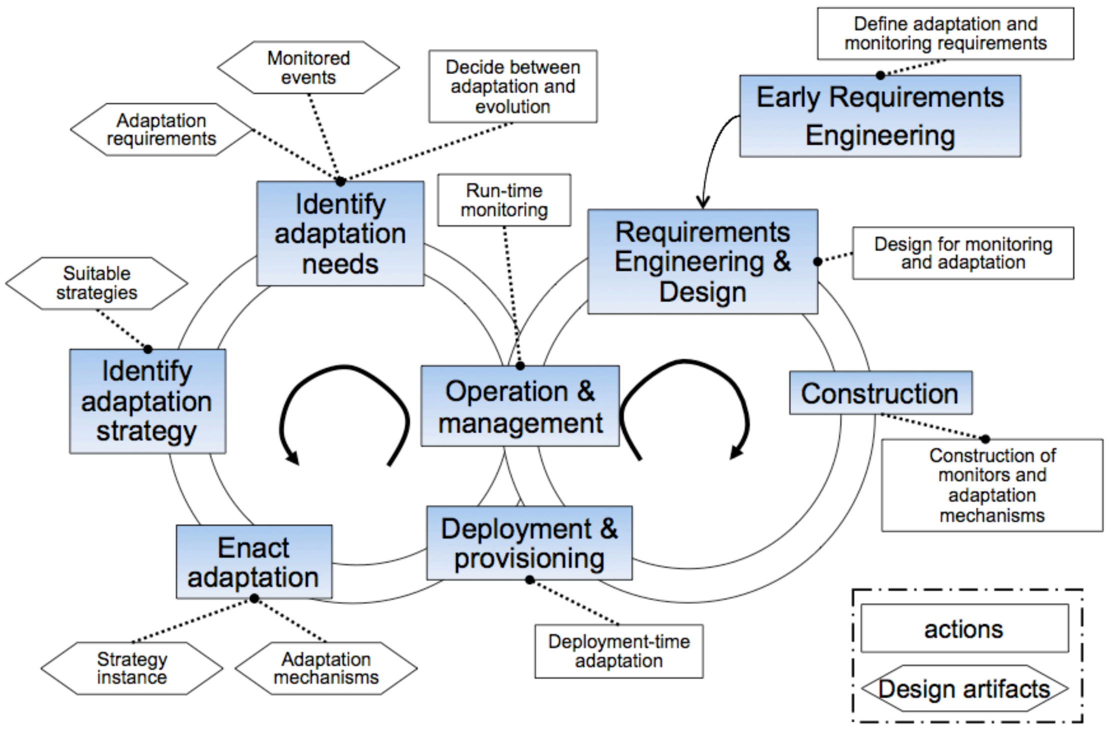

Monitored events
Decide between adaptation and evolution
Define adaptation and monitoring requirements
Early Requirements Engineering
Adaptation requirements
Identify adaptation needs
Run-time monitoring
Requirements Engineering & Design
Design for monitoring and adaptation
Suitable strategies
Identify adaptation strategy
Operation & management
Construction
Construction of monitors and adaptation mechanisms
Enact adaptation
Deployment & provisioning
Strategy instance
Adaptation mechanisms
Deployment-time adaptation
actions
Design artifacts

Fig. 1. Lifecycle for adaptable service-oriented systems.

In the (Early) Requirement Engineering and Design phase, the relevant context dimensions and the application and system characteristics are considered in order to elicitate adaptation and monitoring requirements and in particular define the types of changes that trigger self-adaptation. Subsequently, during the Construction phase, the corresponding monitoring and adaptation mechanisms are designed, developed and then refined until the Deployment and Provisioning phase. At runtime (Operation and Management phase), the system is continuously monitored in order to support the detection of the relevant context and system changes. When changes occur, the system might re- quire evolution or adaptation interventions. Evolution is performed if the system needs to be redesigned and thus it requires the reiteration of the described cycle starting from the requirements engineering and design phase.

In the Identify adaptation need phase, specific situations that demand for adaptation are identified. Each adaptation need has to be associated with particular Adaptation Strategies that are able to satisfy the corresponding adaptation requirements. Based on the current situation, the knowledge obtained from previous executions, and the avail- able adaptation strategies, a reasoner (e.g., a resource management system) selects the most suitable adaptation strategy that will be performed in the Enact adaptation phase. Fig. 1 highlights for each phase the various adaptation- and monitoring-specific actions (boxes) carried out throughout the lifetime of an SBA and the main design artifacts that are exploited to perform adaptation (hexagons).

3 ROIA Development and Execution on Clouds

In Real-Time Online Interactive Applications (ROIA), there are typically multiple users who concurrently access a common application state and interact with each other within one virtual environment. The users access the application from different client machines and control their avatars that can influence other users' avatars. Since ROIA have very high performance requirements, the application state processing is performed on multiple servers: the virtual environment is divided into disjoint areas (zones) and each server is processing clients inside a particular zone, i.e., the overall workload of the application is distributed on multiple resources. Hence, ROIA are highly distributed applications with challenging QoS demands, such as: short response times (about 0.1- 1.5 s), high update rate of the application (up to 50 Hz), large and frequently changing number of users in a single application instance (up to 10_4 simultaneously).

For ROIA, a particular challenge is that the number of users participating in a ROIA session and, thus, the workload, is often subject to daytime-dependent changes. A negative consequence of this are expensive up-front investments to build a suitable server pool which is able to handle peak user numbers but will be underutilized most of the time when the load is below the peak. Hence, dynamic adaptation of application sessions during runtime is crucial for ROIA.

We address this challenge by using Cloud Computing for resource provision. Cloud Computing allows to add/remove resources on demand and promises a potentially un- limited scalability by distributing frequent state computations on an arbitrary number of resources given suitable adaptation mechanisms. Despite a variable number of users, Cloud resources can be used efficiently if the application provides suitable adaptation mechanisms. Hence, using Cloud Computing for resource provision and the Lifecycle Model for implementing adaptable ROIA complement each other.

In order to support ROIA development and adaptation on Clouds, we develop the RTF-RMS resource management system [16] on top of the Real-Time Framework (RTF) [17]. RTF-RMS implements the following mechanisms for ROIA development on Clouds:

1. Monitoring of application-specific data, e.g., update rate, number of entities, etc.
2. Distributionhandlingforthedynamicadaptationofapplicationsessionsbyadding/removing Cloud resources using particular adaptation strategies (described below).
3. Application profiles for implementation of application-specific adaptation triggers.
4. High-level development tools for communication and application state distribution.

4 Using the Lifecycle Model for ROIA development on Clouds

This section describes how RTF-RMS supports the developer in designing adaptable ROIA according to the different phases of the Lifecycle Model. In [2], we showed how the Lifecycle described in Section 2 can be applied for ROIA development. In this section, we demonstrate how the Lifecycle Model is used for ROIA development in Cloud environments by exploiting RTF-RMS.

The design of an adaptive application requires the definition of application and adaptation requirements, suitable adaptation strategies and adaptation triggers. In the following, we illustrate how RTF-RMS supports the developer in designing adaptable ROIA in Cloud environments according to the different phases of the Lifecycle model.

In the “Requirement Engineering” phase, the application developer must identify suitable adaptation requirements for his application. For ROIA, the mechanisms for monitoring and adaptation should be non-intrusive, i.e., take place in parallel with the application execution, such that users are not aware of changes inside the application.

For the “Construction” phase, RTF-RMS provides the developer with a C++ library of high-level functions for optimized communication handling (client-server and inter- server) and efficient application state distribution in a multi-server environment. By using RTF-RMS for communication and distribution handling, monitoring mechanisms are automatically integrated in the application processing. These monitoring mechanisms are used in the next phase of the Lifecycle to implement adaptation triggers.

In the “Deployment and Provisioning” phase, trigger rules for each adaptation trigger are defined. We describe the implementation of adaptation triggers in Section 4.1.

In the “Operation and Management” phase, the application is running and monitoring data are checked continuously against the trigger rules to detect changes in the context or in the system that could require adaptation.

In the “Identify adaptation need” phase, RTF-RMS detects a violation of trigger rules which indicates the demand for adaptation.

In the “Identify adaptation strategy” phase, RTF-RMS analyzes the number of application servers and their current workload to choose an adaptation strategy. A detailed description of adaptation strategies provided by RTF-RMS is given in Section 4.2.

In the “Enact adaptation” phase, RTF-RMS enacts the chosen adaptation strategy and changes the distribution of the application processing accordingly.

It is still possible to consider the application requirements defined in [2]: functional requirements require the correct execution of the application while non-functional requirements are related to the management of resources for a high and variable number of users and their frequent interactions. For adaptation requirements, we identified the demand for transparent, proactive, efficient and autonomous adaptation.

4.1 Adaptation triggers for ROIA on Clouds

In general, the adaptation is triggered when some changes occurs. Such changes may affect the component services or the context of the considered application. ROIA require an adaptation when one of the following changes occurs:

1. Change in Quality of Service: QoS violations may be caused by unreliable Cloud resources. QoS violations for ROIA may be related to changes in update rate, response time, throughput, resource usage, packet loss and service availability.

2. Change in computational context: application requirements sometimes change on the basis of variations of the computational context such as variations of CPU and memory load or incoming/outgoing bandwidth.

3. Change in business context: it refers to changes in user preferences which were not predicted in advance, e.g., too many concurrent users connected to the application or the increase of the number of requests per application.

Listing 1. Excerpt from an example application profile for a fast-paced action game.

```
<appProfile >
     <metric>UpdateRate</metric>
     <addResourceThreshold>25</addResourceThreshold>
     <removeResourceThreshold>100</removeResourceThreshold>
</appProfile >
```

RTF-RMS provides a generic mechanism to implement adaptation triggers by specifying an application profile. In the application profile, developers can specify significant monitoring values for their particular application, e.g., update rate in Hz, see Listing 1 for an example. For each monitoring value in the application profile thresholds have to be defined. If a monitoring value exceeds the `addResourceThreshold`, a new resource is added to the application processing; if monitoring values of any application server fall below the `removeResourceThreshold` dispensable resources are removed. Applica- tion developers can find suitable values for these thresholds by considering the runtime behaviour of their application on physical resources and calculating the virtualization overhead of Cloud resources, or by conducting benchmark experiments in a Cloud.

In order to choose between different adaptation strategies, RTF-RMS creates re- ports for each application server. A report describes the load of a particular server, e.g., current update rate in Hz. A zone report is created for each zone by collecting reports from all servers involved in the zone processing and calculating their average load.

Fig. 2 illustrates an example of choosing adaptation strategies using zone reports. We assigned Server A to the processing of Zone 1, and Server B and C were assigned to Zone 2. The zone report of Zone 1 identifies an average update rate of 20 Hz. Given an `addResourceThreshold` of 25 Hz, RTF-RMS decides to add a new resource to the processing of Zone 1. The zone report of Zone 2 identifies an average update rate of 60 Hz which is between `addResourceThreshold` and `removeResourceThreshold`, but Server B has an update rate of 20 Hz. Hence, RTF-RMS migrates users from Server B to Server C to distribute the workload equally on both servers.

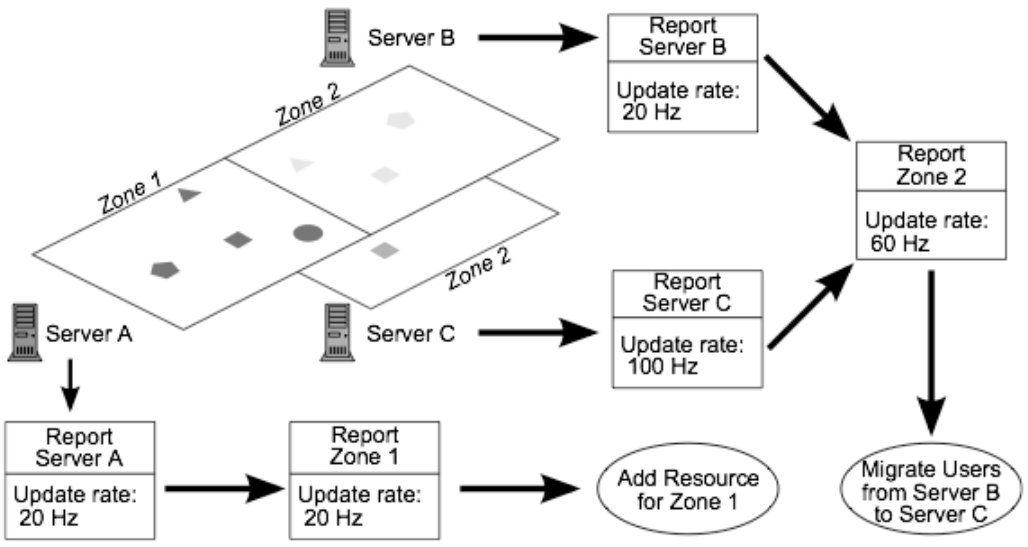


Fig. 2. Finding a suitable adaptation strategy using zone reports.

Another challenge for the implementation of adaptation triggers on Clouds is the compensation of long startup time of Cloud resources which may take up to several minutes. In RTF-RMS, multiple requests for new Cloud resources are sent in parallel to the Cloud API in order to start multiple resources as quickly as possible. Moreover, RTF- RMS introduces a resource buffer to which a predefined number of Cloud resources are moved in advance, i.e. before they are demanded by the application. Resources in the resource buffer are immediately available. If any resource from the resource buffer is integrated in the application processing, RTF-RMS checks whether new Cloud resources must be started in order to keep a certain number of resources in the resource buffer. A detailed description of how to choose the resource buffer size in order to minimize the cost-overhead generated by leasing resources in advance is provided in [16].

4.2 Adaptation strategies for ROIA on Clouds

The adaptation triggers described in the previous section identify the need for adaptation which is implemented by different adaptation strategies. In order to react to changes and avoid inefficiencies, it is necessary to identify the most suitable adaptation strategy that is able to align the application behaviour with the context and system requirements.

Fig. 3 illustrates how RTF-RMS chooses between four adaptation strategies proposed in our previous work [18]:

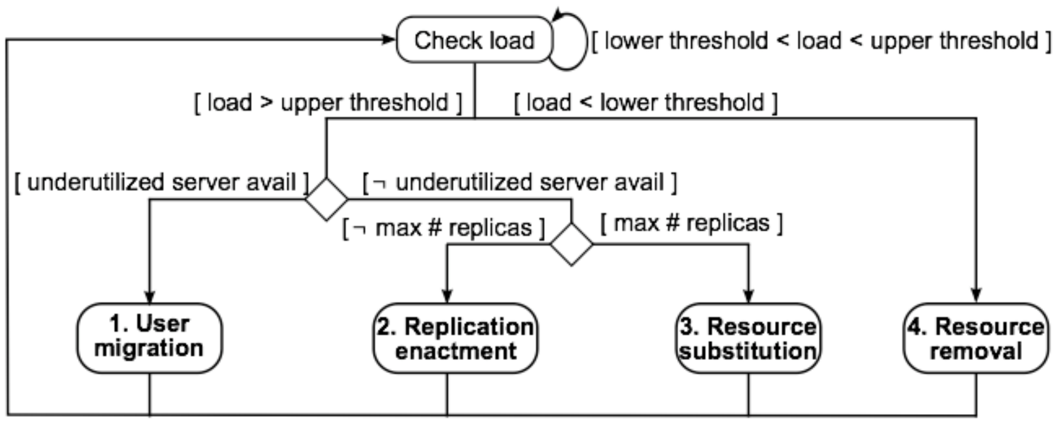

Check load
[lower threshold < load < upper threshold]
[load > upper threshold]
[load < lower threshold]
[underutilized server avail]
[¬ underutilized server avail]
[¬ max # replicas]
[max # replicas]
1. User migration
2. Replication enactment
3. Resource substitution
4. Resource removal

Fig. 3. RTF-RMS chooses between four different adaptation strategies.

1. User migration: Users are migrated from an overloaded server to an underutilized server which is replicating the same zone. For this purpose, user connections are switched from one server to another. RTF-RMS distributes users by default equally between the application servers for a particular zone. User migration is restricted to servers that are replicating the same zone because managing users located in different zones on the same server would considerably increase the inter-server communication for processing user interactions which are typically limited to nearby users. User migration is the preferred action if the load of an overloaded server can be compensated by running resources.

2. Replication enactment: New application servers are added in order to provide more computation power to the highly frequented zone. This strategy is called replication: each application server keeps a complete copy of the application state, but each server is responsible for computing a disjoint subset of entities. As soon as the new server has been added, RTF-RMS migrates a number of users to the new replica in order to balance the load. Replication implies an additional inter-server communication, so its scalability is limited. Since the replication overhead depends on the inter-server communication in a particular application, the maximum number of replicas per zone can be specified in the application profile. If the number of active replicas for a particular zone is below the maximum number of replicas

specified by the application profile, then replication is used; otherwise the resource substitution strategy (described next) is preferred.

3. Resource substitution: An existing application server is substituted by a more powerful resource in order to increase the computation power for highly frequented zones. For this purpose, RTF-RMS replicates the targeted zone on the new resource and migrates all clients to the new server. The substituted server is shut down.

If no better resources are available for substitution, the application reached a critical user density, i.e., large numbers of users in the same geographical area, which cannot be further improved by the generic adaptation strategies offered by RTF- RMS. In this case, the application requires redesign according to the design time activities of the Lifecycle Model.

4. Resource removal: If the update rate of an underutilized application server falls below the `removeResourceThreshold`, RTF-RMS checks whether the zone that is managed by this server is replicated by other servers. If not, nothing happens since each zone must be assigned to at least one application server. If other application servers are replicating this zone, users are migrated equally to these servers, after which the underutilized server is shut down.

Another challenge for the cost-efficient adaptation of ROIA on Clouds is the consideration of leasing periods. In commercial Cloud systems, resources are typically leased and paid per hour or some longer leasing period. Since ROIA have dynamically changing user numbers, the time after which Cloud resources become dispensable is very variable. However, resources will not be used cost-efficiently if they are shut down before the end of their leasing period. In RTF-RMS, resources that have become dispensable are removed from the application processing and moved to the resource buffer. Cloud resources in the buffer are shut down at the end of their leasing period or they are integrated in the application processing again if required.

5 Experiments

In the following, we target the “Enact adaptation” phase of the Lifecycle Model and report experimental results of the replication enactment adaptation strategy using an example of a multi-player action game called RTFDemo [17]. In our experiments, we evaluate how RTF-RMS triggers adaptation if QoS changes as described in Section 4.1; from the adaptation triggers described in Section 4.1, we chose the update rate to trigger adaptation. In order to provide a seamless gaming experience, users should not receive less than 25 updates per second over a longer time period. Hence, we defined an adaptation trigger rule with 25 updates per second as the `addResourceThreshold`.

In our experiments, we use a private Cloud with the Eucalyptus framework (version 2.0.2) [19]; servers are Intel Core Duo PCs with 2.66 GHz and 2 GB of RAM.

We start a single RTFDemo server on a Cloud resource and connect 260 clients to it. Fig. 4 shows that the update rate of Server 1 initially drops with the growing number of connected clients.

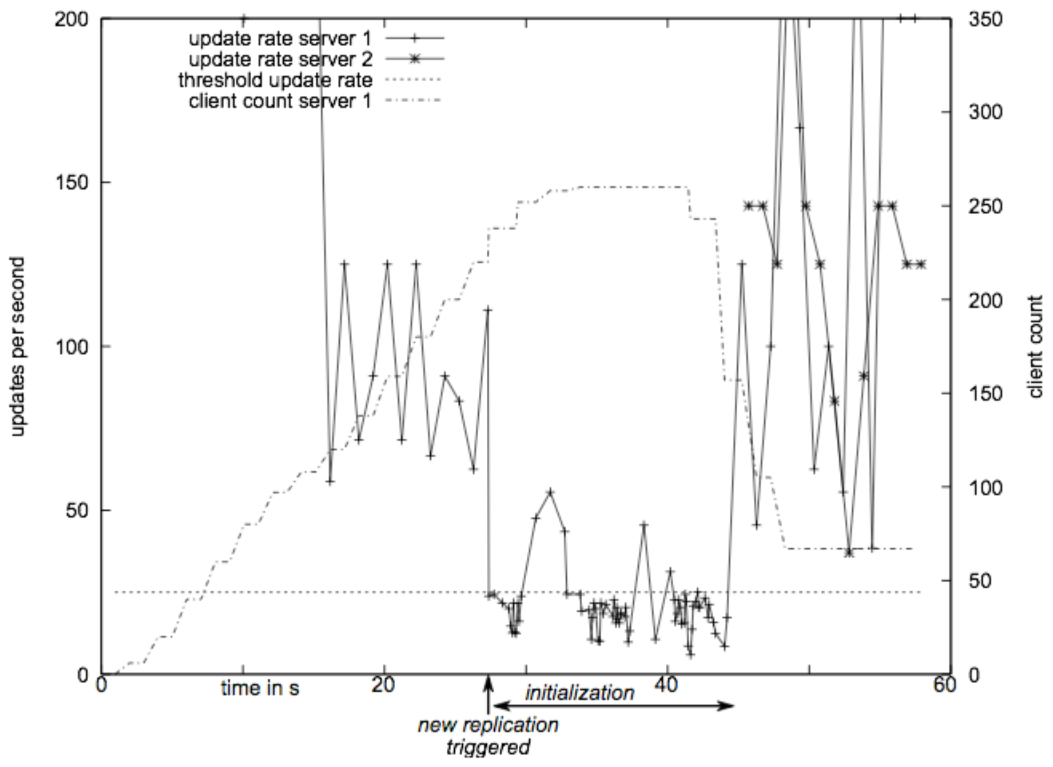


Fig. 4. Load balancing by replication enactment.

When the update rate of Server 1 falls below the threshold of 25 Hz, RTF-RMS requests a new Cloud resource from the resource buffer (for this experiment, the size of the buffer was configured as 1). Although the requested resource is already started up (since it is in the buffer), we observe that a certain time period is still required to add the new server to the application processing and start user migration. This delay of approximately 15 seconds is caused by the initialization and inter-server communication that are required to integrate the new server in the application processing. After the migration is accomplished, the update rate of Server 1 has increased from 20 Hz to about 100 Hz. The update rate of server 1 and server 2 fluctuates between 50 and 200 Hz caused by the continously changing number of interactions between the 260 clients. Note that if the resource were not in the buffer and would have been started up from scratch, the delay would be much longer, in the order of 130 seconds. Hence, using the resource buffer for starting Cloud resources in advance reduces startup times by a factor of approximately 9 which allows for faster adaptation enactment in contrast to solutions that start resources from scratch, e.g., Amazon Elastic Load Balancing [20].

6 Conclusion

This paper presents how the S-Cube Lifecycle Model can be utilized for developing adaptive ROIA on emerging Cloud systems. We showed how our resource management system RTF-RMS implements concrete mechanisms for ROIA development and execution on Clouds according to the Lifecycle. In extension of our previous work that proved the feasibility of applying the S-Cube Lifecycle on ROIA development on a static set of physical resources [2], this paper targets the specific challenges related to Clouds. In particular, we illustrated how the Cloud influences the definition of adaptation triggers and strategies and how RTF-RMS allows for a cost-effective leasing of Cloud resources on demand by buffering unused resources; thereby the startup times of Cloud resources are reduced. Our adaptation triggers are based on application-specific monitoring values provided by RTF-RMS, and, hence, go beyond the state-of-the-art adaptation mechanisms on common Cloud platforms that are based on generic system information. Our experimental results demonstrate how the replication enactment adaptation strategy implemented in RTF-RMS improves the performance of a multi-player, real-time online game and how the resource buffer decreases startup times of Cloud resources.

Acknowledgments: We are grateful to the anonymous reviewers for their helpful re- marks on the preliminary version of the paper. Our research has received funding from the EC's 7th Framework Programme under grant agreement 215483 (S-Cube).

References

[1] "The S-Cube project." http://www.s-cube-network.eu, 2011.
[2] D. Meiländer, S. Gorlatch, C. Cappiello, V. Mazza, R. Kazhamiakin, and A. Bucchiarone, "Using a Lifecycle Model for Developing and Executing Adaptable Interactive Distributed Applications," in Towards a Service-Based Internet, vol. 6481 of Lecture Notes in Computer Science, pp. 175–186, Springer, 2010.
[3] Rational, "Rational unified process - best practices for software development teams," in Tech. Rep. TP026B, 1998.
[4] M. P. Papazoglou and W. v. d. Heuvel, "Service-oriented design and development methodology," in Int. J. Web Eng. Technol., vol. 2, no. 4, pp. 412–442, 2006.
[5] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley, "SOMA: a method for developing service-oriented solutions," in IBM Syst. J., vol. 47, pp. 377–396, 2008.
[6] K. Mittal, "Service oriented unified process." http://www.kunalmittal.com/html/soup.html, 2009.
[7] D. Linner, H. Pfeffer, I. Radusch, and S. Steglich, "Biology as inspiration towards a novel service Life-Cycle," in International Conference on Autonomic and Trusted Computing (ATC 2007), p. 94102, 2007.
[8] S. Lane and I. Richardson, "Process models for service based applications: A systematic literature review," in Information and Software Technology, 2010.
[9] Y. Wautelet, Y. Achbany, J. Lange, and M. Kolp, "A process for developing adaptable and open service systems: Application in supply chain management," in International Conference on Enterprise Information Systems (ICEIS 2009), vol. 24, pp. 564–576, Springer, 2009.
[10] S. Vale and S. Hammoudi, "Model driven development of context-aware service oriented architecture," in The 11th IEEE International Conference on Computational Science and Engineering - Workshops, pp. 412–418, 2008.
[11] T. Margaria, B. Steffen, M. Wirsing, et al., "SENSORIA patterns: Augmenting service engineering with formal analysis, transformation and dynamicity," in Leveraging Applications of Formal Methods, Verification and Validation, Third International Symposium, vol. 17, pp. 170–190, 2008.
[12] G. Spanoudakis, A. Zisman, and A. Kozlenkov, "A service discovery framework for service centric systems," in Services Computing, 2005 IEEE International Conference on, vol. 1, pp. 251–259, 2005.
[13] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, and Z. Wu, "The METEOR-S approach for configuring and executing dynamic web processes," in Tech. Rep., 2005.
[14] A. Bucchiarone, C. Cappiello, E. di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore, "Design for adaptation of Service-Based applications: Main issues and requirements," in Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, vol. 6275 of Lecture Notes in Computer Science, pp. 467–476, Springerg, 2009.
[15] S. Lane, A. Bucchiarone, and I. Richardson, "SOAdapt: A Process Reference Model for Developing Adaptable Service-Based Applications," in Information and Software Technology, 2011.
[16] D. Meiländer, A. Ploss, F. Glinka, and S. Gorlatch, "A Dynamic Resource Management System for Real-Time Online Applications on Clouds," in Lecture Notes in Computer Science, Springer, 2011. To appear.
[17] "The Real-Time-Framework (RTF)." http://www.real-time-framework.com, 2011.
[18] F. Glinka, A. Raed, S. Gorlatch, and A. Ploss, "A Service-Oriented Interface for Highly Interactive Distributed Applications," in Euro-Par 2009 – Parallel Processing Workshops, vol. 6043 of Lecture Notes in Computer Science, pp. 266–277, Springer, 2010.
[19] D. Nurmi, R. Wolski, C. Grzegorczyk, et al., "The Eucalyptus Open-Source Cloud-Computing System," in 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131, IEEE Computer Society, 2009.
[20] "Amazon Web Services." http://aws.amazon.com, 2011.

Appendix B: Addressing highly dynamic changes in service-oriented systems: Towards agile evolution and adaptation

Andreas Metzger and Elisabetta Di Nitto

Abstract. This chapter sets out to introduce relevant foundations concerning evolution and adaptation of service-oriented systems. It starts by sketching the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service-oriented systems. Then, it provides an overview and more thorough explanation of the various kinds of changes that may need to be faced by service-oriented systems. To understand how such changes could be addressed, the chapter introduces a reference service life-cycle model which distinguishes between evolution, viz. the manual modification of the specification and implementation of the system during design-time, and (self-)adaptation, viz. the autonomous modification of a service-oriented system during operation. Based on the discussion of the key activities prescribed by that life-cycle, the chapter elaborates on the need for agility in both adaptation and evolution of service-oriented systems.

1 Introduction

For future software systems and software development processes, the only constant will be change. The \world" in which those future software systems operate is reaching unprecedented levels of dynamicity [12, 14]. Those systems will need to operate correctly in spite of changes in, for example, user requirements, le- gal regulations, and market opportunities. They will have to operate despite a constantly changing context that includes, for instance, usage settings, locality, end-user devices, network connectivity and computing resources (such as offered by Cloud computing). Furthermore, expectations by end-users concerning the personalization and customization of those systems will become increasingly relevant for market success [2].

Modern software technology has enabled us to build software systems with a high degree of flexibility. The most important development in this direction is the concept of service and the Service-oriented Architecture (SOA) paradigm [15, 22, 21]. A service-oriented system is built by \composing" software services (and is thus also called \service composition" or \composed service" in the literature). Software services achieve the aforementioned high degree of flexibility by separating ownership, maintenance and operation from the use of the software. Service users do not need to acquire, deploy and run software, because they can access its functionality from remote through service interfaces. Ownership, maintenance and operation of the software remains with the service provider [14].

While service-orientation offers huge benefits in terms of flexibility, service- oriented systems face yet another level of change and dynamism. Services might disappear or change without the user of the service having control over such a change.

Agility, i.e., the ability to quickly and effectively respond to changes, will thus play an ever increasing role for future software systems to live in the highly dynamic \world" as sketched above. Agility can be considered from two view- points:

- First, agility may concern the evolution of the system. This means that it concerns the development process and how engineering activities (such as requirements engineering and implementation) should be performed to timely address changes by evolving the software.

- Secondly, agility may concern the adaptation of the system. This means that it concerns the system itself and how the system should respond to changes [29]. Agility in adaptation is typically achieved through self-adaptation, i.e., the autonomous modification of a service-oriented system during operation.

In this chapter, we first sketch the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service- oriented systems (Section 2). Then, we provide an overview and more thorough explanation of the various kinds of changes that need to be faced and how these could be addressed (Section 3). As reference for the remainder of the chapter, we then introduce a service life-cycle model which integrates evolution and adaptation into a coherent framework (Section 4). After elaborating on the activities prescribed by that life-cycle, we discuss the need for agility in evolution (Section 5) and adaptation (Section 6). We conclude this chapter with our perspectives on agile development for service-oriented systems (Section 7).

2 Historical Development

2.1 The Emergence of the SOA Paradigm

In [14] we gave an extensive account of the historical development of software technology and methods toward highly dynamic, service-oriented systems. The following paragraphs briefly summarize the major milestones along this development.

“Genesis": In the late 1960ies software development processes started to get disciplined through the identification of well-defined stages and criteria, which were to be met in order to progress from one stage of the process to the next. The waterfall life-cycle model as proposed by Royce in 1970 [30] was such an attempt. It was very rigid and advocated the need for software developers to focus not only on coding but also on higher-level activities (requirements analysis and specification, as well as software design) and on verification and validation. At the time those life-cycle models were defined, the \world" was assumed to be relatively fixed and static. Stable requirements could thus be elicited at the beginning of the development process. Additionally, most organizations were monolithic. Accordingly, solutions addressing their requirements were to a large extent also monolithic and centralized.

"Enlightenment": It was soon realized that the assumption about the stability of requirements was not realistic. In most practical cases, requirements cannot be fully gathered upfront and then left untouched [27]. Specifically, it was realized that often stakeholders do not know what they exactly expect from a system beforehand. Changing requirements should consequently be considered as an intrinsic factor that must be dealt with during development. As a consequence, incremental and prototyping-based life-cycle models emerged. These were introduced to achieve better tailoring of solutions to requirements and to mitigate the risks involved in software development. "Industrialization": The development of software technology soon allowed dynamic bindings among modules and { even more importantly { to extend these bindings across network boundaries (examples include CORBA and Java RMI). This allowed for the distributed execution of the software. The development of software technology was accompanied by an increased automation of software development activities. This included, for instance, automatic verification techniques and tools which reached a level of maturity that allowed them to be applied to real-life problems. Examples include model checkers or Boolean satisfiability checkers.

"Globalization": Another major development followed regarding the ownership of software. In the beginning, software development was under the control of a single organization which ultimately owned the code completely. Then, component-based software development became dominant. O_-the-shelf components were developed and provided by third parties who were also responsible for their quality and their evolution. Software development thus became (partly) decentralized.

The development of software technology and methods further made it possible to support seamless evolution of the software in order to incorporate certain anticipated changes. These included, for instance, additional or redefined module functionality. However, as motivated in Section 1, the demand for software to continuously respond to highly dynamic changes of its context and its requirements has reached unprecedented levels in the past few years. A further major step in the development of software technology and methods to address this dynamism was the birth of the service concept and the Service-Oriented Architecture (SOA) paradigm [15, 22, 21].

2.2 What is SOA?

When referring to Service Oriented Architecture (SOA) as a paradigm, SOA typically constitutes a set of guiding principles for building service-based applications. Thanks to these principles, services can be (re-)used in many difierent settings and service-based applications can meet the requirements for dynamism and flexibility.

A detailed discussions of the SOA principles can be found in [15, 22, 21], they include:

- Loose coupling: This principle means that a service only makes weak assumptions about its interactions with other services. For example, instead of services being tightly coupled by means of a common data model (such as was the case for distributed objects and CORBA), a small set of simple data types is used. The loose coupling principle of SOA also argues for preferring asynchronous interactions over synchronous ones, message passing instead of method invocation, as well as flexible message routing instead of fixed. Finally, late binding during deployment or even run-time (see below) is preferred over static binding during design-time.

- Dynamic service discovery and late binding: This principle implies that ser- vices can be discovered and composed into a service-oriented during deployment and during run-time. The discovery of services is supported by service registries (\yellow pages") or even more powerful service search engines (see [28] for more details on the latter aspect). Those facilities allow for dynamically re-configuring a service-based application by replacing services, which may have shown to be unreliable, with alternative compatible services (possibly from different service providers).

- Service interoperability and protocol independence: Similar to W3C's notion of Web Service, a software service may be considered a \piece" of software designed to support interoperable machine-to-machine interaction over the Internet. Existing Internet and Web Service standards allow for service inter- operability by prescribing ways how services can interact, exchange messages and be located across the Internet. Those standards have lead to the proliferation of services available over the Internet and described in terms of WSDL (the Web Services Description Language). As an example, at the time of writing, the seekda.com search engine has indexing almost 30.000 software services.

- Self-containment and autonomy of services: Self-containment and autonomy means that the logic that is governed by a service resides within an explicit boundary. The service should only have control within this boundary and should not depend on other services for it to execute its governance. This allows for keeping changes and failures isolated and will foster reusability of services.

- Abstraction and service interfaces: The functionality that is exposed by a service naturally abstracts from the underlying implementation. In addition, the only part of a service that is visible to the outside world is what is exposed via the service interface. Typically, the underlying logic of the soft- ware, which is irrelevant to the service user and/or constitutes intellectual property of the service provider, will not exposed via the service interface. As a consequence, this means that the service interfaces define a form of formal contract. In order to interact with the service, the service users only need to have knowledge about the service interface (and not the underlying logic or implementation). This principle thus is an application of the well-known \information hiding" principle to the realm of service-based systems. It should be noted that a service-based application or a software service does not necessarily need to follow all of the aforementioned principles. There may be good reasons that in certain application scenarios and domains, some of the principles are not applicable. As an example, for more traditional enterprise applications it may well suffice to have static service bindings and a more tight coupling between those services by sharing more complex data structures.

2.3 The Impact of SOA

SOA enables us to build software systems with a high degree of flexibility. Soft- ware services separate ownership, maintenance and operation from the use of the software. Service users thus do not need to acquire, deploy and run software, because they can access its functionality from remote through service interfaces. Ownership, maintenance and operation of the software remains with the service provider [14].

Similar to what has been enabled by globalization in the real world, third- party software services now enable organizations to flexibly outsource business functions (typically commodity functions) and to focus on the innovative functions, which differentiates one organization from another.

Thus, SOA promises huge benefits in terms of dynamism and flexibility. However, service-based applications also need to become resilient to their services changing, disappearing or violating their expected quality. Especially in the case of third-party services, the service users do not have control over such changes, thus, calling for novel solutions to address this new dimension of changes. Luckily, the introduction of SOA technology was accompanied by another major step in the development of techniques for automating software engineering activities. Verification and measurement techniques have started to be extended to the operation phase of the systems, leading to “online" techniques [12], including monitoring [28], online testing [5] and run-time verification [6]. Those techniques provide the foundation for service-oriented systems to automatically identify and respond to certain changes (see Section 3). As an example, by continuously verifying whether the service-oriented system meets its requirements, an adaptation of the system can be triggered once a failure becomes imminent.

3 Changes

Adaptation and evolution of a service-oriented system are triggered by changes that may occur in three major areas: (1) the expectations that its users (or other stakeholders) have concerning the functionality and quality that the sys- tem should provide, (2) the world in which the software system is executed, (3) the system itself. The first area is the realm of requirements, the second one is traditionally called context, and the third one is often referred to as the machine [20].

According to the IEEE Standard Glossary of Software Engineering Terminology [19], \a requirement is: (1) a condition or capability needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) a documented representation of a condition or capability as in (1) or (2)".

The term context derives from the Latin cum (with or together) and texere (to weave). It has been defined by Dey and Abowd [13] as "any information that can be used to characterize the situation of entities (persons, places, objects) that are relevant to the interaction between a user and an application, including the user and the application themselves". According to Hofer et al., context can be [17] physical, i.e., measured by some hardware sensor or logical, i.e., captured by monitoring user interaction. When it is physical it refers to location, movement, and any environmental information. When it is logical it refers to users' goals, emotional state, business processes, etc.

Figure 1 shows typical examples of changes that may occur and trigger the need for adapting or evolving the service-oriented system.

Fig. 1. Areas for Changes and Examples

Clearly, the examples of changes shown in the figures require different levels of intervention on the corresponding software system and its artefacts. For instance, the lack of computational resources may be addressed easily, without modifying the structure of the software system, by exploiting the flexibility offered by Cloud computing [4]. The failure of a service is also simple to address if our system is built according to the SOA paradigm that enables dynamic binding (see Section 2) of alternative and compatible services. In this case, in fact, the substitution of one service for another service can occur at runtime without performing any reprogramming activity of the software [26].

Other kinds of changes, such as the addition of new features, require a deeper intervention on the software system. In this case the system may need to be partially redesigned and reimplemented in order to address the change. All changes that can be addressed by performing some simple reasoning (e.g., \since the user is in downtown Milano, he/she is certainly interested in knowing about free parking spaces in that area") or by modifying the bindings to services can be addressed by a service-oriented system if it incorporates self-adaptation facilities [12]. Vice versa, the changes that require redesign or reimplementation of the system typically have to be addressed by the intervention of human beings.

The boundary between the two kinds of changes is not necessarily defined once and for all but depends on the reasoning abilities that we are able to include in the software system while we build it. This point will be discussed in more detail in Section 4.

4 Service Life-Cycle Model

This section introduces the service life-cycle model, as defined by S-Cube, the European Network of Excellence on software, services and systems3. The S-Cube service life-cycle model [28, 25] defines the relevant activities for self-adaptive service-oriented systems and integrates these into a coherent framework. Self- adaptive systems automatically and dynamically adapt to certain changes (see Section 3). The aim of self-adaptation is to reduce the need of human intervention as much as possible in order to allow those systems to quickly respond to changes. The S-Cube service life-cycle model consists of the following two loops, which we will detail in the remainder of this chapter. The loops of the life-cycle can be executed in an incremental and iterative fashion, as follows:

- The Evolution loop (see right hand side of Figure 2) builds on the more traditional development and deployment activities, including requirements engineering, design, realization, and deployment (see Section 5.1). However, it extends those activities with \design-for-adaptation" steps, such as to define and implement how the system should monitor and modify itself when entering the left-hand side of the life-cycle (e.g., see [12, 7]).

- The Adaptation loop (see left hand side of Figure 2) explicitly defines activities for autonomously addressing changes during the operation of service- oriented systems (see Section 6.1). The activities in the adaptation loop follow the steps of the MAPE loop (Monitor-Analyze-Plan-Execute), which is typically found in autonomic systems [31].

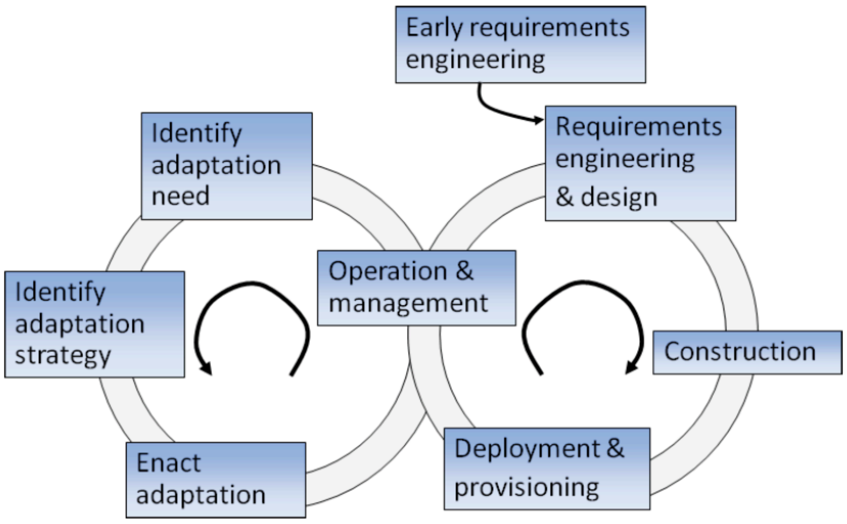

Early requirements engineering
Identify adaptation need
Requirements engineering & design
Operation & management
Identify adaptation strategy
Construction
Enact adaptation
Deployment & provisioning

Fig. 2. The S-Cube Service Life-Cycle Model

It should be noted that in some cases also adaptation requires human intervention. This is often called human-in-the-loop adaptation [28]. Human-in-the- loop adaptation is different from evolution in the sense that the activities performed by the humans and the artifacts that are modified differ; e.g., the change of requirements documents certainly requires to go through the evolution loop, while the choice between two possible candidate services can be performed as human-in-the-loop adaptation.

5 Evolution

When the system evolves, it goes through a reenginering phase in which it is permanently modified. The practices used in this phase are being deeply studied in the software maintenance literature (see for instance the proceedings of the 26th International Conference on Software Maintenance [18]). Some of these practices have been applied to service-oriented systems as well. In this case, the main aspects which have been considered concern the issues of maintaining the interface compatibility between versions of services, of identifying the right timing for evolving services depending on the contracts currently in place, of identifying the right stakeholders for evolution and of understanding their impact to the evolution process (see [3] for a detailed overview).

5.1 Activities in the Evolution Loop

As mentioned above, the activities in the evolution loop follow the traditional software development activities.

Requirements Engineering: During requirements engineering, the functional and quality requirements for the service-oriented system are elicited and documented. The specifics of service-oriented systems make requirements engineering a particularly relevant activity. This is related to the highly dynamic nature of service-oriented systems and to the necessity to realize the continuous adaptability of these systems. Indeed, in a context where the application is in continuous evolution and is thus characterized by rather blurred boundaries, the study of those requirements that exist a priori in the organizational and business setting and that are hence largely independent from the solution becomes very important [9].

Design: During the design phase, the workflow of the service-oriented system is typically specified using languages such as BPEL. Together with the definition of the workflow, candidate services are identified that can provide the functionality and quality to fulfill the requirements of the system. This means that those services that provide, at least partially, the expected functionality and quality are identified. This is supported by service matchmaking techniques, such as the ones presented in [11]. A further task is to define adaptation strategies and mechanisms which enable the application to react to adaptation needs [7], i.e., to take \design for adaptation" decisions. Finally, the conditions under which some changes will have to be enacted during the runtime have to be identified.

Realization: After design, the realization and implementation of the sys- tem can start. This specifically means that contracts on quality aspects (aka. Service Level Agreements, SLAs) have to be established with the third-party service providers. Typically, this requires some form of SLA negotiation and agreement [28, 11]. Moreover, the monitoring and analysis mechanisms needed to identify the conditions that require changes at runtime have to be defined together with the adaptation strategies to be executed.

Deployment: Deployment comprises all the activities needed to make the system available to its users, including the deployment of internal services and software components on computing infrastructures (including Clouds [4]). It should be noted that the service-oriented systems itself could be offered as a service and could thus participate in other service compositions.

5.2 Towards Agile Evolution

An important element of the evolution loop is efficiency. In the age of globalization, in fact, changes need to be handled in a timely fashion. If the execution of the evolution loop is not able to quickly address changes, there is a risk of delivering a solution that addresses a certain change when this change is not relevant any longer, for instance, because it has been superseded by other new changes. Agile development approaches [1, 10] are often mentioned as a way to address changes in a fast and interactive way, if developers are able to work in close collaboration with the owners of new requirements or with those who have a deep knowledge about the occurred context changes.

This statement has an initial evidence in the work of Capiluppi et al. [8] where an empirical study has been conducted which shows how the application of agile methods to support the evolution of a software system has resulted in a "smooth evolution while avoiding the problems of increasing complexity or decreasing customer satisfaction".

In the context of SOA, companies such as IBM [23] and OutSystems [33] are suggesting the adoption of an agile approach. [23] argues that refactoring is an important technique in SOA, where services and service interfaces need to be continuously adjusted to the needs of new customers. The paper also observes that management of agility in SOA is simplified by the fact that changes tend to be localized in specific parts of software systems and, in particular, in the service choreography (aka. service composition).

The adoption of an agile approach supported by a toolset called the \Agile Service Platform" is proposed in [33]. Among other features, this platform keeps information about components and dependencies continuously updated, thus effectively supporting the maintenance and the evolution activities, for which a 70% reduction in costs has been reported. Interaction with customers during the development and the evolution phases is strongly encouraged - facilitated by the platfom by means of creating proper communication channels between customers and the project management.

A technological advancement that pushes agile development of SOA to the extreme is offered by mashups [34]. Mashups constitute the integration of different web applications and services which have the purpose of serving the specific needs of some users. They are usually short lived systems intended to be built not only by expert developers but also by less-experienced people. To this end, proper development environments are being developed. These offer a specific component and composition models and are usually associated to some runtime environment. While these environments promise to shorten the development cycle in a significant way, they are still not well integrated into a proper full-edged development methodology.

From a completely different perspective, another interesting step toward agility is the integration of service selection within the requirement engineering phase suggested by Maiden et al. [35]. They have proposed a tool called EDDIE that supports engineers in the definition of requirements and use cases, as well as in the identification of available services that fulfill such requirements with different levels of precision.

Even though it has not been designed with agile processes in mind, agile methods could be mapped to the S-Cube life-cycle. Using Scrum [32] as an example, we could assume that each iteration of the evolution loop is performed as a Sprint which aims at delivering an increment of a service-oriented system. Given that services represent a natural unit of functionality, each such Sprint could aim at incorporating a new service in the service-oriented system up to the point where the complete functionality is offered to the system's users. Still, as the aim is to build self-adaptive systems, an agile approach targeted at service-oriented systems needs to take into account that the redesign and redevelopment of the software system has to incorporate the \design for adaptation" principle (see Section 5.1) in a seamless way. This is an issue which surely deserves further research.

6 Adaptation

6.1 Activities in the Adaptation Loop

As mentioned above, the activities in the adaptation loop follow the MAPE loop as known from autonomic computing.

Operation and Management: Operation and management include all activities needed for running and controlling the service-oriented system. The literature also uses the term governance to describe all activities that oversee the correct execution of service-oriented system (and its constituent services). The identification of changes, including problems in the running system (e.g., failures of constituent services) and alteration of its context, plays a fundamental role. This identification is obtained by means of monitoring mechanism and, more generally, by exploiting techniques for run-time quality assurance such as online testing or run-time verification [12, 28]. Together, these mechanisms and techniques are able to detect relevant changes.

Identify Adaptation Need: Certain changes trigger the service-oriented system to leave its \normal" operation and enter the adaptation or the evolution cycle. As discussed in Section 3, which of the two loops is being entered depends on the kind of observed change. The adaptation cycle is responsible for autonomously deciding whether and how the service-oriented system needs to modify itself in order to maintain its expected functionality and quality.

Identify Adaptation Strategy: When the adaptation loop is entered, possible adaptation strategies are identified, selected and instantiated. Possible strategies include service substitution (rebinding), SLA re-negotiation, and re-configuration of the workflow [7]. From the set of possible strategies, the ones which _t the situation are selected and instantiated, e.g., by deciding which ser- vice to use as a substitute or which re-configuration of the workflow to perform.

Enact Adaptation: After the adaptation strategy has been selected and instantiated, adaptation mechanisms are used to execute the actual adaptation. For example, service substitution, re-configuration or re-composition may be obtained using automated service discovery and dynamic binding mechanisms, while re-composition may be achieved using existing automated service composition techniques.

6.2 Towards Agile Adaptation

In addition to the degree of automation, the point in time when a change can be detected impacts on the agility of the system in responding to that change; e.g., if the system can forecast an imminent change, more time and thus more options remain for the adaptation than if the system can only detect changes once they actually occur.

The following three types of adaptation [24] exemplify the impact that the point in time when changes are detected has on the agility of the system:

- Reactive adaptation refers to the case in which the system is modified in response to external failures that have actually occurred, i.e., failures that are actually observed by the users of the system. Repair and/or compensation activities have to be executed as part of the adaptation in order to mitigate the effects of the failure; e.g., the user is paid a compensation, or the workflow is rolled-back. Obviously, such a reactive adaptation can have a severe impact on how agile a system can respond to changes [16]. As examples, the execution of reactive adaptation activities on the running system can considerably increase execution time and therefore reduce the overall performance of the running system, or an adaptation of the system might not be possible at all, e.g., because the system has already terminated in an inconsistent state.

- Preventive adaptation refers to the case in which an actual local failure or deviation is repaired before its consequences become visible to the user in the form of an external failure. As an example, if a local failure occurs (such as a third-party service S1 responding too slow), the system might forecast whether this may lead to an external failure (visible by the user) and prepare repair mechanisms; e.g., a faster service S30 could be used instead of service S3 in the remainder of the workflow, thereby counteracting the slow response of service S1. Due to the time delay between the detection of the local failure of S1 and the observation of the external failure by the user, there is more flexibility in modifying the workflow than in the reactive case.

- Proactive adaptation refers to the case in which the system is modified even before a local failure occurs. As an example, if the system is able to predict that a local failure is imminent (but did not yet occur), the system can be flexibly modified in advance; e.g., a service S1, predicted to respond too slow, can be replaced by a quicker service S10. In this case, neither repair nor compensation activities would be necessary as part of the adaptation, as no failure has actually occurred.

In a nusthell, the more agile the service-oriented system is to become, the stronger the role of proactiveness in the adaptation loop becomes. This means that already during design (i.e., within the evolution loop), decisions need to be made about how to predict failures during the execution of the service-oriented system. However, selecting the right technique can be quite a challenging task due to the highly dynamic nature of service-oriented systems (see [24] for an in-depth discussion on this issue).

7 Conclusion and Research Highlights

Agile methods applied in the SOA context are expected to lead to the quick and effective development and evolution of self-adaptive service-oriented systems. Self-adaptation has a strong implication on the way the agile development process is organized. Taking Scrum as an example, Sprints should not only concern the inclusion of new functionalities but also the creation of respective self-adaptation mechanisms. The challenge is therefore to understand how self-adaptation mechanisms can be developed in an incremental way, Sprint by Sprint.

Also, Sprints are more effective when associated to the creation of business value. It is thus important to identify the value that can be associated with a specific self-adaptation capability that allows the system to reliably work in a globalized environment and to execute resiliently under highly dynamic changes. How to perform such a quantification is still open and should certainly be part of future research.

Acknowledgments

Research leading to these results has received funding from the European Community's 7th Framework Programme FP7 / 2007-2013 under grant agreement 215483 (S-Cube).

References

1. P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen. New directions on agile methods: a comparative analysis. In Proceedings of the 25th International Conference on Software Engineering, ICSE '03, pages 244{254, Washington, DC, USA, 2003. IEEE Computer Society.
2. G. Adomavicius and A. Tuzhilin. Personalization technologies: a process-oriented perspective. Commun. ACM, 48:83{90, October 2005.
3. V. Andrikopoulos. A Theory and Model for the Evolution of Software Services. Tilburg University, The Netherlands, 2010.
4. M. Armbrust, A. Fox, R. Gri_th, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. Commun. ACM, 53:50{58, April 2010.
5. A. Bertolino. Software testing research: Achievements, challenges, dreams. In FOSE '07: 2007 Future of Software Engineering, pages 85{103, Washington, DC, USA, 2007. IEEE Computer Society.
6. D. Bianculli, C. Ghezzi, and C. Pautasso. Embedding continuous lifelong veri_cation in service life cycles. In Principles of Engineering Service Oriented Systems (PESOS 2009), co-located with ICSE 2009, 2009.
7. A. Bucchiarone, C. Cappiello, E. D. Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore. Design for adaptation of service-based applications: Main issues and requirements. In Engineering Service-Oriented Applications: Supporting Software Service Development Lifecycles (WESOA) co-located with ICSOC, ServiceWave, 2009.
8. A. Capiluppi, J. Fernandez-Ramil, J. Higman, H. Sharp, and N. Smith. An empirical study of the evolution of an agile-developed software system. In Software Engineering, 2007. ICSE 2007. 29th International Conference on, pages 511{518, may 2007.
9. B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. Mller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. Software engineering for self-adaptive systems: A research roadmap. In B. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, editors, Software Engineering for Self-Adaptive Systems, volume 5525 of Lecture Notes in Computer Science, pages 1{26. Springer Berlin / Heidelberg, 2009.
10. D. Cohen, M. Lindvall, and P. Costa. An introduction to agile methods. Advances in Computers, pages 1{66, 2004.
11. M. Comuzzi and B. Pernici. A framework for QoS-based web service contracting. ACM Transactions on web, 3(3), 2009.
12. R. de Lemos, H. Giese, H. M□uller, M. Shaw, J. Andersson, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cikic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. M. Goeschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, M. Litoiu, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezz_e, C. Prehofer, W. Sch□afer, W. Schlichting, B. Schmerl, D. B. Smith, J. P. Sousa, G. Tamura, L. Tahvildari, N. M. Villegas, T. Vogel, D. Weyns, K. Wong, and J. Wuttke. Software Engineering for Self-Adpaptive Systems: A second Research Roadmap. In R. de Lemos, H. Giese, H. M□uller, and M. Shaw, editors, Software Engineering for Self-Adaptive Systems, number 10431 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

13. A. K. Dey and G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In Workshop on The What, Who, Where, When, and How of Context-Awareness, 2000.
14. E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl. A journey to highly dynamic, self-adaptive service-based applications. Automated Software Engineering, 2008.
15. T. Erl. Service-oriented Architecture. Prentice Hall, 2004.
16. J. Hielscher, R. Kazhamiakin, A. Metzger, and M. Pistore. A framework for proactive self-adaptation of service-based applications based on online testing. In Ser-viceWave 2008, number 5377 in LNCS. Springer, 10-13 December 2008.
17. T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, and J. Altmann. Contextawareness on mobile devices: the Hydrogen approach. In 36th Annual Hawaii International Conference on System Sciences, pages 292{302, 2002.
18. 26th IEEE International Conference on Software Maintenance (ICSM 2010), September 12-18, 2010, Timisoara, Romania. IEEE Computer Society, 2010.
19. Institute of Electrical and Electronics Engineers, New York. IEEE Standard Glos-sary of Software Engineering Terminology, IEEE Std 610.12-1990 edition, 1990.
20. M. Jackson. Problem Frames: Analysing and Structuring Software Development Problems. Addison-Wesley, New York, 2001.
21. N. Josuttis. SOA in Practice: The Art of Distributed System Design. O'Reilly Media, 2007.
22. D. Kaye. Loosely Coupled: The Missing Pieces of Web Services. RDS Press, 2003.
23. P. Krogdahl, G. Luef, and C. Steindl. Service-oriented agility: An initial analysis for the use of agile methods for SOA development. In IEEE International Confer-ence on Services Computing, pages 93{100, Los Alamitos, CA, USA, 2005. IEEE Computer Society.
24. A. Metzger. Towards accurate failure prediction for the proactive adaptation of service-oriented systems (invited paper). In Proceedings Workshop on Assurances for Self-Adaptive Systems (ASAS), collocated with ESEC 2011, 2011.
25. A. Metzger, E. Schmieders, C. Cappiello, E. D. Nitto, R. Kazhamiakin, B. Pernici, and M. Pistore. Towards proactive adaptation: A journey along the s-cube service life-cycle. In Maintenance and Evolution of Service-Oriented Systems, 2010.
26. O. Moser, F. Rosenberg, and S. Dustdar. Non-intrusive monitoring and service adaptation for WS-BPEL. In Proceeding of the 17th international conference on World Wide Web, WWW '08, pages 815{824, New York, NY, USA, 2008. ACM.
27. B. Nuseibeh. Weaving together requirements and architectures. IEEE Computer, 34(3):115{117, 2001.
28. M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, editors. Service Research Challenges and Solutions for the Future Internet: Towards Mechanisms and Meth-ods for Engineering, Managing, and Adapting Service-Based Systems. Springer, Heidelberg, Germany, 2010.
29. M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. IEEE Computer, 40(11):38{ 45, 2007.
30. W. Royce. Managing the development of large software systems. In IEEE WESCON, pages 1{9, San Francisco, CA, USA, 1970.
31. M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, 4(2), 2009.
32. K. Schwaber. Agile project management with Scrum, volume 7. Microsoft Press Redmond (Washington), 2004.
33. D. Sprott. Product Overview: OutSystems Agile SOA Platform. CBDI Journal, pages 15{20, April 2009.
34. J. Yu, B. Benatallah, F. Casati, and F. Daniel. Understanding mashup development. IEEE Internet Computing, 12(5):44{52, 2008.
35. K. Zachos and N. Maiden. Inventing requirements from software: An empirical

investigation with web services. In Proceedings 16th IEEE International Conference on Requirements Engineering, pages 145{154. IEEE Computer Society Press, 2008.

Appendix C: Managing evolving services

Michael P. Papazoglou, Vasilios Andrikopoulos, Salima Benbernou

INTRODUCTION

Services are subject to constant change and variation, leading to a continuous service re-design and improvement effort. Service changes originate from different sources such as introducing new functionalities to an existing service, modifying the current functionality of a service in order to improve its performance, or including new regulatory constraints requiring a change of the behavior of services. Such changes lead to a continuous service re-design and improvement effort. However, they should not be disruptive by requiring radical modifications in the very fabric of services, or the way that business is conducted between service providers and consumers.

With the term *service evolution* we refer to the *continuous process of development of a service through a series of consistent and unambiguous changes* [Papazoglou2008]. The evolution of services is expressed through the creation and decommissioning of *service versions* during their lifetime. These versions have to be aligned with each other in a non-disruptive manner and in a way that allows a service designer to track the various modifications and their effects on the service in terms of consistency. Looking at the effect that service changes have on their consumers in the Service Oriented Architecture (SOA) paradigm, we can classify them as *shallow* and *deep*. Shallow changes are incremental changes, localized to one service and restricted to the consumers of the service. Such incremental changes are based on the notion of *compatibility* between service versions to allow for a seamless and transparent update of the service to the service clients. Shallow changes therefore require a robust versioning scheme and an unambiguous definition of compatibility in order to be managed efficiently. Deep changes, on the other hand, are transformational changes, cascading beyond the clients of the service and potentially to entire end-to-end service networks. As such, they require a change-oriented service development methodology that considers the scope, effect, effort and applicability of such changes in across enterprise environments and throughout entire service chains.

SERVICE AND SOFTWARE EVOLUTION

Evolution in software systems has been traditionally considered as either a part, or a synonym of *software maintenance*. The insight gained by early studies resulted in empirical laws that drive and govern the evolution of software systems. Evolution is particularly important in distributed systems, and therefore for service-oriented ones too, due to a complex web of software component interdependencies. As Bennet and Rajlich point out [Bennett2000], attempting to apply conventional maintenance procedures (halt, modify and re-execute) in large distributed systems (like the ones emerging in SOA) is not practical for a number of reasons:

- Identifying which services constitute the emerging system is non-trivial, especially in the context of large service networks.
- Access to the actual source code of third-party services is limited or not existent at all due to the strong encapsulation and loosely coupled properties enforced by SOA. Many well-known maintenance techniques like refactoring or impact analysis are therefore very difficult or even impossible to apply.

Towards this direction, Bennet and Rajlich decompose maintenance into *evolution* and *servicing* and treat the former as an iterative development phase, and the latter as the more traditional post-development corrective, perfective and preventive actions. The rest of this discussion follows this distinction and emphasizes the continuous development aspect of evolving services.

EVOLUTION IN COMPONENT-BASED SYSTEMS

Historically and conceptually, Component-Based Systems (CBS) can be considered a predecessor of SOA, which expands and builds on the same principles of encapsulation, independence and unambiguous definition of interfaces. Component evolution is a well-documented field with established techniques stemming from Software Configuration Management (SCM). Evolving a component (or a service), includes addressing changes in both its interfaces and its implementation, with each one of these aspects having different evolutionary requirements. Due to their capacity for composition and emphasis on reuse, components exhibit strong dependencies with other components that they consume. Changing a component may therefore have implications, often unforeseen ones, to other components. Upgrading to a new component may require for both versions (old and new) to be deployed in parallel while the transition takes place. Finally, identifying and distinguishing between different versions of components requires the introduction of SCM techniques like version identifiers incorporated into e.g. the component meta-data. Since version identifiers do not explain what changes occurred between versions, checking for compatibility with different component versions has usually to be performed separately.

A crucial difference between component and service evolution however is that, while in the former an older version of a component can continue to be used if necessary, despite it being deprecated by the component developer, this is not possible in the latter case. This is because services and service-based applications rely on invoking remotely other services during their execution, rather than including them as libraries in their executables. Replacing an older service version with a new, incompatible one will always result in breaking the clients of the services who cannot fall back on using the older version since it is not available anywhere anymore. Furthermore, components and services are quite different in terms of coupling and binding types, invocation methods, and composition mechanisms [Papazoglou2011]. While, therefore, many of the lessons learned by component evolution can be reused in the context of service evolution, there are also significant differences between them.

SERVICE ADAPTATION

Adaptation is one of the mechanisms proposed in the literature for dealing with service evolution. Service adaptation is initiated by a change either in the context of the service (consumer requirements, laws and regulations, market dynamics, corporate strategy) or the service itself (re-design, technological advancements). It involves different mechanisms for adapting either the interface or the implementation of the service (or both) to the interoperability requirements of the service consumers. Interface adaptation focuses on solving the mismatches in the signature and/or communication protocol of interacting services by modifying the interfaces accordingly. Some kind of adapter is usually involved, which is generated in the general case in a semi-automatic manner, and based on the parametric transformation of the interfaces that are expected by the service consumers and the actually provided ones by the service. Interface adapters can also be layered on top of each other, allowing service developers to provide (ideally) a unique implementation endpoint that exposes multiple versions of interfaces that are mapped to each other with adapters, instead of multiple versions of the service. The maintenance cost then is moved to ensuring the consistency and efficiency of the layering of the adapters and out of the service life cycle itself.

Implementation adaptation on the other hand focuses on composite services, i.e. services comprised of aggregations of other services, in many cases defined as orchestrations in Business Process Execution Language (BPEL). As in the case of interface adaptation, most approaches use semi-automatic transformations between different versions, usually included as a set of predefined adaptation scenarios (dealing with the “known unknowns”). Special provision is taken for replacing services when they are unavailable or in some way unsuitable for use, e.g. by dropping below an acceptable Quality of Service (QoS) level, with other equivalent ones. Dynamic binding mechanisms are used in this case to ensure that the service endpoints invoked by the composite service are changed without disrupting the operation of the service.

The application of service adaptation techniques however, both for interface and composition, is not always possible without explicit manual intervention. In this sense, these approaches are limited in their automation. They may be successful in preserving interoperability with a desired set of consumers, but by definition they require a number of modifications towards this purpose. These modifications may in turn interfere with the operation of other services by the same organization in terms of resources (computational and financial) and code. The benefit of adaptation in a resource-centric environment like enterprise services should therefore always be weighted first against its cost.

In any case, all existing approaches for service evolution, based on adaption and not, require implicitly or explicitly the use of a compatibility definition between service providers and consumers in order to ensure the interoperability between them. For this purpose in the following section we present a set of theories and techniques for managing service compatibility.

SERVICE COMPATIBILITY

Interface changes affect the ability of service providers and consumer to exchange messages between them. Ideally, such changes should allow the two parties to be able to exchange (valid and acceptable) messages, despite any interface changes that may happen to either side. In order to assure that this exchange is possible, we define on the notion of service version compatibility. Service version compatibility guarantees that we can introduce a new version of either a provider or a client of service messages without changing the other [Papazoglou2011]. Compatibility is classified in two dimensions:

- *Horizontal compatibility,* or interoperability, of two services[1]. Horizontal compatibility expresses the requirements that allow two or more services to participate successfully in an interaction, either as service producers or service consumers, in at least one context under which the services can fulfill their roles.
- *Vertical compatibility* or substitutability/replaceability (from the provider's or the consumer's perspective, respectively) of service versions. Vertical compatibility expresses the requirements that allow the replacement of one version of a service by another in a given context.

In both cases, context is defined as a configuration of the environment in terms of the execution state of both service producer and service consumer, along with the status of their resources, and for a particular message exchange history [Andrikopoulos2011]. Inverting the traditional viewpoint on compatibility, there are two types of changes to a service definition that guarantee version compatibility:

- *Backward compatibility:* a new version of a message client is introduced and the message providers are unaffected. The client may introduce new features but should still be able to support all the old ones.
- *Forward compatibility:* a new version of a message provider is introduced and the message clients that are only aware of the original version are unaffected. The provider may have new features but should not add them in a way that breaks any old clients.

Some changes are both backward- and forward-compatible; for example the addition of a new service method to an existing service description does not affect its existing consumers if everything else remains the same. In such cases we talk about full compatibility, or simply compatibility. Full compatibility allows the replacement of an existing service version with an equivalent (that is, compatible) one without any effect on the correct operation and performance of its clients.

From a practical standpoint, compatible service evolution in the services description standard Web Services Description Language (WSDL) is limited to service changes that are either backward or forward compatible, or both. The types of service changes that are compatible are:

1. Addition of new (WSDL) operations to an existing document.
2. Addition of new XML schema data types in a WSDL document, if and only if they do not affect existing types.

Incompatible change types on the other hand include most of the modifications possible on a service interface: removing an operation, renaming an operation, changing the parameters (in data type or

[1] We use the term services here in its most general sense, denoting any two or more parties interacting using the service-oriented paradigm. The definition therefore applies to both the relation between a composite service and its constituent (composite or not) services, and that of a service-based application and the services it relies on. A similar assumption applies also to the following definitions.

order) of an operation and changing the structure of a complex data type. In [Andrikopoulos2011], an alternative approach is discussed for enabling the compatible evolution of services. Instead of restricting service changes to the short list above, a theoretical framework is presented which allows for reasoning on the evolution of services. As a result, further compatible changes (called *T-shaped*) are allowed; for example, removing data elements from incoming message data types and adding data elements in outgoing message data types.

Service version compatibility for changes to the (structural) signatures of the service interfaces is based on two fundamental premises of type theory, providing a direct connection with object-oriented programming languages and practices:

- *Service argument contra-variance*: if the argument of a service is redefined, the new argument types must always be an extension (generalization) of the original ones.
- *Service result co-variance*: if the result of a service is redefined, the new result types must always be a restriction (specialization) of the original ones.

When evolving a business protocol, usually defined in the BPEL language, states and transitions may be added to or removed from an active protocol. A new version of a protocol is created each time its internal structure or external behavior changes. The perception that clients have of a specific protocol is called a protocol view. Since the client's view of a protocol is restricted only to the parts of the protocol that directly involve the client, a client might have equivalent views on different protocols. Clients whose views on the original and target protocols are the same are essentially not affected by evolution. Protocol compatibility aims at assessing whether two protocols can interact, i.e. if it is possible to have a conversation between the two services despite changes to their protocols. Compatibility of two protocols can be either *complete*, i.e. all conversations of one protocol can be understood by the other protocol, or *partial*, when there is at least one conversation possible between the two protocols.

SERVICE VERSIONING

A robust versioning strategy is needed to allow for upgrades and improvements to be made to a service, while continuously supporting previously released versions. Service versioning is therefore an important issue for service developers and providers alike [Papazoglou2011]. Versioning as a concept has its roots in the SCM field, which has contributed in major ways to software maintenance and evolution [Bennett2000]. From the different mechanisms developed under SCM, of particular interest for service evolution is that of development support using versioning as summarized in [Estublier2005]. Versioning in this context refers to the keeping of historical records of the various software artifacts in the domain of responsibility as they undergo change. The reliance of services on publishing their service interface descriptions (in the majority of cases in one of the versions of the WSDL language) adds an additional dimension to the versioning of services, on top of the traditional one requiring the versioning of their implementing components. More specifically, it requires the promotion of structured documents to first-class software objects that need to be versioned and related to the other objects (documentation, code, test-related documents). These service interface description documents are the only means of interaction with the service and confine for that purpose the executable code to an internal to the service role. Interface versioning therefore is essential for allowing services to evolve over time.

VERSIONING TECHNIQUES

Service interface version techniques reuse the methods developed and used in practice for a number of years in the context of SCM. In particular, versioning approaches also distinguish between major and minor releases. In the context of services, a major release signifies an incompatible service version, while minor releases assume the existence of (vertical) compatibility between the new and older version. Uniquely identifying a service version in the version space usually entails including a MajorNumber.MinorNumber identifier in the service interface. Version ID "1.2" for example, denotes the 2nd minor revision of the 1st major release (which has version ID "1.0"). Alternatively, the naming scheme may incorporate a release date stamp.

Service interfaces are in the vast majority of cases defined as documents of the XML dialect known as the WSDL language. It follows naturally therefore that version identification should be applicable to XML documents. Along these lines, version identifier information in practice is included directly in the XML namespace of the document and/or the namespace of its data types. Using a new namespace inadvertently results in disrupting the binding of the service on the consumer side, i.e. it "breaks" the existing consumers of the service. A similar result is achieved by incorporating the version identifier in the endpoint URL of the service interface. Both methods are therefore meant to be used only if a major version of a service is deployed.

An alternative, more flexible and informative approach is to include dedicated version identifier attributes in the root element of the WSDL document and/or to each element of the document (in the case of allowing different identifiers for the data types and interface signature definitions). Despite its usefulness, this method is not supported by the WSDL specification. This, in turn, requires the consumers to be somehow able to process the versioning information and understand the implications of the naming scheme on application level. Lack of consensus on how this information is supposed to be handled on a lower, communication level, and the absence of standardization efforts mean that such a solution can only be used if the service consumer applications are specifically designed to incorporate them. This generates a degree of coupling that goes contrary to the loose coupling nature of SOA; it may be acceptable in some cases however.

VERSIONING STRATEGIES

In the case of service evolution, the cost of provisioning for multiple service interfaces is non-linear [Papazoglou2011]. The development of a new service interface requires additional effort in binding the interface versions with the underlying implementation(s). Since, as we discussed above, many changes to the service interfaces lead to incompatible service versions, service providers in practice need to support multiple active (i.e. non-decommissioned) versions of the same service, as shown in Fig. 1. Otherwise they need a) to notify their service consumers about the applied changes (which in many cases it is not possible, e.g. in the case of public services that do not use Service Level Agreements (SLAs) with their consumers), and b) to rely on the service consumers to adapt to the new service version (instead of moving to another service provider offering an equivalent service). Offering multiple active service versions, however, additionally requires access to a number of resources in the supporting infrastructure for each active version (e.g. computational resources in the service container, connections to databases, storage space for its transactional logs). Furthermore, each version adds managerial overhead in terms of monitoring and auditing in order to ensure that its operation complies with the agreed-upon Service Level Agreements (SLAs) with the provider's customers.

FIGURE 1

Providing for multiple active versions can therefore be overtaxing for the service provider. A balance between the cost of losing customers to the competition due to inconsistent updates to the service, and that of maintaining many versions of the service at a time, is usually reached by minimizing the amount of active versions through the use of compatible changes. The best practice for this purpose relies on a compatibility-oriented strategy for versioning: maintain multiple active service versions for major releases (i.e. incompatible versions), but cut maintenance costs by grouping all minor releases under the latest (compatible) one [Andrikopoulos2011], as shown in Fig. 1. Special provision has to be taken for the decommissioning period of versions to be deprecated. Jerijaervi and Dubrais [Jerijaervi2008] discuss different approaches for decommissioning versions. Usually, a "grace" period is given before the version becomes inactive, notifying the service consumers if an SLA has been signed with them, or they are subscribed to a dedicated informational service which can push version change notifications to them. This is simply however an industrial best practice which has not been standardized and therefore it relies on service developers to enforce it.

CHANGE-ORIENTED SERVICE LIFECYCLE

Managing deep service changes requires a *change-oriented service life cycle* methodology to provide a sound foundation for spreading changes in an orderly fashion so that impacted services in a service-chain are appropriately (re-)configured, aligned and controlled as the changes occur [Papazoglou2011]. The purpose of the change-oriented service life cycle is to ensure that standardized methods and procedures are used for the efficient and prompt handling of all service changes, in order to minimize the impact of change-related incidents upon service operation and quality. This means that in addition to functional (structural and behavioural) changes, a change-oriented service life cycle must deal with policy-induced, operational and non-functional changes. The objective is to achieve actual end-to-end QoS capabilities for end-to-end services to achieve the proper levels of service required. Toward this goal, the lifecycle focuses on ensuring that services are performing as desired, and that, out-of-control or out-of-specification conditions are anticipated and responded appropriately. This includes traditional QoS capabilities, e.g. security, availability, accessibility, integrity and transactionality, as well as service volumes (e.g. number of service events, number of items consumed, service revenue) and velocities (i.e. its performance characteristics). The combination of these measurements provides all the information needed to understand how an enterprise is performing in terms of its services.

FIGURE 2 – the life-cycle phases

Fig.2 illustrates such a deep change-oriented service life cycle that comprises of a set of inter-related phases, activities and tasks that define the change process from the start through to completion. Each phase produces a major deliverable that contributes towards achieving change objectives. The phases of the life cycle are discussed in the following.

PHASES OF THE CHANGE-ORIENTED LIFECYCLE

As shown in Fig. 2, there are four phases in the lifecycle: need to evolve, analyse impact of changes, align-refine-define and operational service. The initial phase (need to evolve) focuses on identifying the need for change and scoping its extent. One of the key activities in this phase is identifying the root causes of the need for change and their potential implications. For instance, compliance to regulations is one of the major forces for change and may lead to the transformation of all services within a service network. The impacted individual services in the end-to-end service (called services-in-scope) therefore need to be identified. In addition, service performance metrics, such as Key Performance Indicators (KPIs), need to be collected, both for the services-in-scope and for the end-to-end service. The information collected in this phase (change causes, scope, services-in-scope and KPIs) is given as input in the following phase in the cycle.

The second phase (analyse impact of changes) focuses on the actual analysis, re-design and improvement of the existing services, and the estimation of the cost of applying the proposed changes. The ultimate objective of this phase is to provide an in-depth understanding of the functionality, scope, reuse, and granularity of the services-in-scope that are identified for change in the previous phase. The problem however lies in determining the difference between existing and future service functionality, and assessing the impact of the transition to the proposed functionality. For this purpose, instead of applying the changes directly on operational services, organizations rely on the existence of “as-is” and “to-be” service models. Analysts rely on an “as-is” service model to understand the portfolio of available services. This model is used as the basis for a comprehensive re-engineering analysis of the current portfolio of available services that need to evolve. The “to-be” service model is used as the basis for describing the target service functionality and performance levels after applying the required changes.

To determine the differences between these two models a *gap analysis model* is used to help prioritize, improve and measure the impact of service changes. Gap analysis is a technique that purposes a services realization strategy by incrementally adding more implementation details to an existing service to bridge the gap between the “as-is” and “to-be” service models. Gap analysis commences with comparing the “as-is” with the “to-be” service functionality to determine differences in terms of service performance (expressed as KPIs) and capabilities. Service capabilities determine whether a process is able to meet specifications, customer requirements, or product tolerances. Service changes may spill over to other services-in-scope in the service chain. It is therefore essential during the analysis to be able to recognize the scope of changes and functionality that is essentially self-sufficient for the purposes of an evolving service. When dealing with deep service changes, several types of problems need to be addressed: service flow problems, service control problems, overlapping and/or conflicting functionality and input/output problems [Papazoglou2011]. With respect to cost estimation, this process involves identifying and weighing all services-in-scope that need to be re-engineered in order to estimate the cost of the re-engineering project. In cases where costs are prohibitive for an in-house implementation, an outsourcing policy might be pursued, or the decision for re-engineering might be reconsidered.

During the third phase (align, refine, define), all the new and changed services are aligned, integrated, tested and when/if found appropriate, put into production. To achieve this, a service integration model is created to facilitate the implementation of the service integration strategy. The service integration model establishes the integration relationships between service consumers and providers, determines the message distribution needs, delivery-responsible parties, and provides a service delivery map. Finally, the service integration model addresses the message and process orchestration needs for the resulting end-to-end service. The resulting service integration strategy includes service design models, policies, SOA governance options, and organizational and industry best practices. The role of the services integration model ends when a new (upgraded) end-to-end service architecture is completely expressed and validated against technological specifications provided by infrastructure, management/monitoring and technical utility services. The resulting service then enters the last phase of the lifecycle (operational service), until the need for change is again identified, initiating once more the process described in the previous phases.

CONCLUSIONS

Despite its connections with component evolution, service evolution poses a number of additional challenges due to the strongly encapsulated and loosely coupled systems (i.e. services) that it deals with. In this context, service compatibility and versioning become important mechanisms for enabling the seamless update of a service without affecting its existing consumers. Such changes are not always possible however. In this case it is required of service developers to consider the scope and impact of the change and weigh the outcome against the effort and resources required for applying it. A systematic change-oriented service lifecycle should be used for this purpose.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

Appendix D: A Variable Context Model for Adaptable Service-Based Applications

Antonio Bucchiarone

Fondazione Bruno Kessler
Via Sommarive, 18, Trento TN 38100, Italy
bucchiarone@fbk.eu

Cinzia Cappiello, Elisabetta Di Nitto, Barbara Pernici and Alessandra Sandonini

Politecnico di Milano, Dipartimento di Elettronica e Informazione
Piazza Leonardo Da Vinci 32 20133 Milano, Italy
{cappiell,dinitto,pernici}@elet.polimi.it

Keywords

Service-based Applications, Context-Awareness, Modeling, and Adaptation

INTRODUCTION

Service-based applications are deployed in dynamic and distributed settings by composing different services, possibly owned by different service providers. System developers exploit the functionality offered by services without having them under their direct control. This introduces critical dependencies between service-based applications and their component services that could change or be unavailable without notice.
Since the developer of the service-based application usually aims at guaranteeing continuity of service to its users, service-based applications have to be equipped with adaptation capabilities to react dynamically and automatically to unavailability of the component services.
Service selection becomes an important aspect of SBAs; such selection depends on the global requirements, on the functionality and on the quality of service to be provided. Moreover, it may depend also on the *context* in which services are executed. Context is defined as: "*any information that can be used to characterize persons, places or objects that are considered relevant to the interaction between a user and an application, including users and applications themselves*" (Dey 2001). Context should be taken into account for service selection. For example, the service that informs a sailorman about the weather forecasts on a specific route should be very detailed and focused on the conditions of the sea and of winds, while the service dedicated to a family willing to decide if to book a trip on the seaside should be focusing on the weather conditions of a specific place, typically, on a longer time scale.
More in general, service-based applications should be able to adapt the execution flow to address changes of the execution context. For example, applications have to be flexible in order to satisfy users' variable requirements on the basis of the situation (e.g., geographical position, time) in which users are when they access the application.

A first goal of this paper is to provide a novel context model for adaptable service-based applications and to point out to its role in the adaptation activities. The context model is the basis for the definition of triggers enabling adaptation or evolution of service-based applications and enables the identification of the information that has to be collected and monitored at run-time. We propose an approach to context modeling which is itself adaptive to the current situation. In fact, the relevant contextual information might be different in different

situations: for instance, the information needed about a location may be different depending on the location (e.g., a small village vs a large town), or depending on the user who is interacting with the application (e.g., a user who knows a location well vs a user who is not familiar at all with the location). In these cases the way the application behaves might vary not only according to the context in general, but according to the representation of the context itself that is variable in the different situations.
Context data are gathered by using different kinds of sensors. An important issue is to understand the level of granularity at which to collect data. The level of granularity is defined by the amount of details to catch for representing the significant characteristics of the environment for a given application. In fact, it is convenient to avoid the collection of unnecessary details of data that are not suitable to catch changes in the execution context. For example, in case of emergency situations, it is necessary to collect data values every second in order to have a fine control of the situation, while in other situations a daily value is enough since a more frequent measure of context data does not provide any additional relevant information. In location-based applications, in some cases, coarse information about the location (e.g. a country) may be sufficient, while in other cases it is important also to be aware of the regional context.

As a result of the previous considerations, another goal of this paper is to address the issue of the granularity of monitored context data and to propose a way to support adaptation, by analyzing context data at different granularities and using a proactive approach to establish the adaptation needs and the dynamic invocation of the services.

The paper is structured as follows. Section 2 describes previous work related to context-based adaptation and highlights the innovative contributions of our approach. Section 3 describes the phases that compose the life cycle for designing service-based applications together with the specific actions to perform and the artifacts exploited to perform adaptation. Section 4 provides details about the framework we propose to enable context-aware adaptation. Capabilities of the presented framework are discussed in Section 5 by means of an example.

RELATED WORK

The goal of this section is to give an overview of approaches and frameworks that are strictly related to what we propose in this paper. In particular we have categorized them in the following three topics: adaptation of SBAs, context-aware frameworks for SBAs and approaches for composition and adaptation of SBAs.

ADAPTATION OF SERVICE-BASED APPLICATIONS

Adaptable systems change their behavior, reconfigure their structure and evolve over time reacting to changes in the operating conditions, so to always meet users' expectations. As suggested in (Ardagna & Pernici 2007), adaptation mechanisms can either be embedded in the description of the adaptable SBAs or implicit in its structure. Various frameworks can be found in the literature with the objective to support adaptation of SBAs, each of them addressing a specific issue. Most of them concern *built-in adaptation*, i.e. the adaptation logic is completely specified at design time. They concentrate on how to specify adaptation mechanisms and adaptable applications, exploiting different tools. For instance, the specification may be performed by extending standard notations (i.e., BPEL) with adaptation specific tools (Karastoyanova et al.,2005) using event-condition-action like rules (Baresi et al., 2007, Colombo et al, 2006), variability modeling (Hallerbach et al, 2008) or aspect-oriented approaches (Kongdenfha et al., 2006).
In the literature there are, however, proposals of frameworks for dynamic adaptation, all featuring an adaptation manager separated from the application. Notably, all these approaches are in the service-oriented field. In (Spanoudakis et al., 2005) the authors consider the problem of adapting the

application by replacing malfunctioning services at runtime. The adaptation rule is fixed at design time, but it is dynamically applied by a manager component that monitors functional and non-functional properties, creates queries for discovering malfunctioning services and replaces them with dynamically discovered replacements.
Narendra et al. (Narendra et al., 2007) proposes an aspect-oriented approach for runtime optimization of nonfunctional QoS measures. The METEOR-S framework (Verma et al., 2005) supports dynamic reconfiguration of processes, based on constraints referring to several QoS dimensions. Reconfiguration is performed essentially at deployment-time. PAWS (Ardagna et al., 2007) is a framework for flexible and adaptive execution of web service based applications. At design-time, flexibility is achieved through a number of mechanisms, i.e., identifying a set of candidate services for each process task, negotiating QoS, specifying quality constraints, and identifying mapping rules for invoking services with different interfaces. The runtime engine exploits the design-time mechanisms to support adaptation during process execution, in terms of selecting the best set of services to execute the process, reacting to a service failure, or preserving the execution when a context change occurs. Finally, the Vienna Runtime Environment for Service-oriented Computing (VRESCo) (Hummer et al., 2011) is a framework implemented to address issues like dynamic selection, binding and invocation of services.

CONTEXT-AWARE FRAMEWORKS

The literature is rich of contributions focusing on the development of context-aware applications. In (Baldauf et al., 2007) an interesting classification of some well-known frameworks is proposed together with an architectural model that is suitable for describing the main elements of a context-aware application.

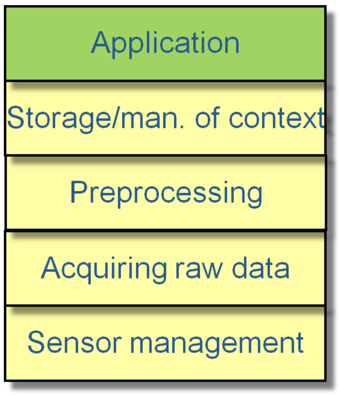


Figure 1. Architectural model of context-aware applications, from (Baldauf et al., 2007).

As shown in Figure 1, this model is structured in five layers. At the lowest layer we have the mechanisms needed to discover and manage sensors of contextual data. On top of this layer it is possible to collect data from these sensors. These data are then preprocessed to extract relevant contextual information from them. The contextual data are stored according to the defined context model and can be queried by application-level components. While it is possible to build an application from scratch developing all layers shown in Figure 1, specific frameworks usually provide support for what concerns the first three or four layers in the figure. Some approaches such as CASS (Fahy & Clarke 2004) offer a centralized context server while others such as Hydrogen (Hofer et al., 2002) assume that the entire stack of layers is to be offered on small devices like smartphones. Interested readers can refer to (Baldauf et al., 2007) for a detailed comparison between the various frameworks.
Other approaches offer a programming model specifically focused on context. For instance, Hirschfeld et al in (Hirschfeld, Costanza & Nierstrasz 2008) present a programming approach where contextual concepts are first class elements. The idea behind the approach is that when, during the execution of a program, a context change

is detected, one or more layers of code are activated/deactivated. Each layer defines a way to change the behavior of a certain piece of code depending on the context change that has occurred. While the approaches mentioned above focus mainly on how to acquire context information and on how to handle the corresponding changes in the software system, the literature on DBMS has been focusing on the problem of determining the proper interval of data sampling to avoid missing information about relevant events and on time granularity. Also, some approaches focusing on the definition of rich context models have been proposed. For instance, in (Cappiello et al. 2006) the problem of modeling several alternative context sub-dimensions (e.g. for space and time) has been studied, but the granularity of such information is not discussed.
In (Bolchini et al 2009) a context model that captures different context dimensions is presented. It is called context dimension tree. An example is shown in Figure 2.

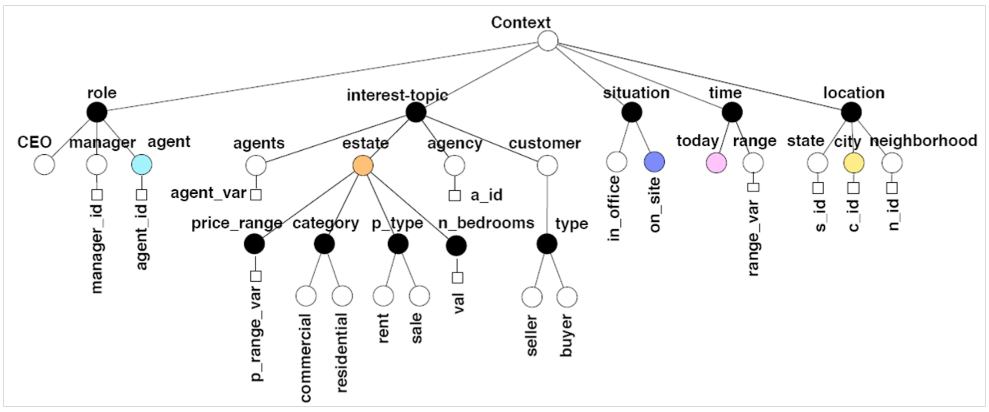


Figure 2. An example of context dimension tree (Bolchini et al 2011).

Besides the relationships explicitly shown by the graphical notation that represent either “part of” or “is alternative to” relationships, other context-validity constraints can be properly defined. Single contexts are defined as subtrees of a context dimension tree, representing the contexts currently envisaged for a particular application. A partial order between more or less abstract contexts can also be defined. This context model has been recently integrated with a framework for acquiring context information from sensors (Camplani et al., 2011).

CONTEXT-AWARENESS COMPOSITION AND ADAPTATION

So far we have presented approaches to adapt SBAs and to acquire, model and manage context in generic pervasive system. The literature offers also approaches focusing on exploiting contextual information within service composition and on providing context-dependent self-adaptation mechanisms.
In (Autili et al., 2007) a conceptual model for context-aware service oriented applications is introduced: it shows the relationships among all the entities constituting an adaptable context-aware service-oriented application. Such conceptual model could be considered as a sort of dictionary for the users and the developers.
(Chaari et al. 2007) propose an architecture for a service based application trying to separate the aspects of context management and adaptation from the service application core. Authors propose to apply adaptation to three different levels in the service application: service (application behaviour), data (content adaptation) and user interface (visualization). Service behaviour can be adapted substituting the current version of a service with a one that could be more suitable for the measured context. The flexibility of such system is limited since all the versions of the services have to be pre-built depending on the different contexts. In (Martin et al. 2006) context-based adaptation for mobile learning systems has been studied: suitable activities are suggested on the basis of the context in which the user is operating. Each user has preferences in terms of needs, interests, personal features and learning styles. The proposed adaptation mechanism is implemented in three steps; each of them is

in charge of the analysis of the information and of the selection of the most suitable activity for a user, depending on his/her context.
The approach that appears to be closest to ours is the one presented in (Niu, Li, Zhao, Tang & Shi 2011), where the idea of a multi-granularity context model is presented, in particular, for location and time values, with an ontological approach to reason on the context and to compose services in the most appropriate way.
The main distinctive element of our approach compared to the others in the literature is that our context model integrates the idea of various levels of granularity for context with the idea of describing as part of the context model the dynamics of transition from a context value to the other. Moreover, the context model is integrated into a highly dynamic self-adaptation approach that is able to compute the proper adaptation policies on the fly, when they are needed.

Summary. From the analysis of the literature, it is possible to notice a lack of a generic architecture able to cover all the aspects behind the adaptation of context-aware applications and of a general context model able to accommodate context variability in the modeling aspects and not only in its values. In this paper the effort would be focused in such direction. In particular, the proposed framework is able to cover all the aspects that are at the basis of the construction and operation of a context-aware adaptable application and would constitute a sort of general environment in which all the approaches in the literature could take place. Our framework enables the construction of an adaptable service based application. Starting from the context modeling, we aim at offering a generic framework able to capture all the relevant aspects behind the development of such applications. In particular we would like to offer a formalization of the main concepts that will be used in the framework for context-aware adaptation. In particular, we focus on the relationships that could exist among the adaptation triggers and the decision about the most suitable adaptation strategy to execute.

A LIFE-CYCLE FOR ADAPTABLE SBAs

Figure 3 shows the life cycle for adaptable SBAs presented in (Bucchiarone et al., 2009) and (Bucchiarone et al ., 2010). It is composed of two cycles: (i) *evolution cycle* that leads to the explicit re-design of an application, (ii) *adaptation cycle* that is performed at run time and addresses all the cases in which the adaptation needs are addressed on-the-fly. The two cycles coexist and support each other during the lifetime of the application. The figure highlights the various adaptation- and monitoring-specific actions (boxes) carried out throughout the life-cycle of the SBA, the main design artifacts that are exploited to perform adaptation (hexagons), and the phases where they are used (dotted lines).

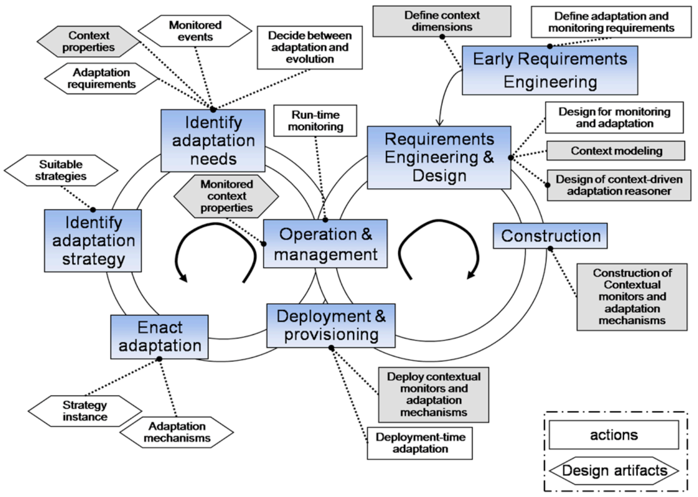

Context properties
Monitored events
Decide between adaptation and evolution
Define context dimensions
Define adaptation and monitoring requirements
Early Requirements Engineering
Adaptation requirements
Identify adaptation needs
Run-time monitoring
Requirements Engineering & Design
Design for monitoring and adaptation
Context modeling
Design of context-driven adaptation reasoner
Suitable strategies
Monitored context properties
Identify adaptation strategy
Operation & management
Construction
Construction of Contextual monitors and adaptation mechanisms
Enact adaptation
Deployment & provisioning
Deploy contextual monitors and adaptation mechanisms
Strategy instance
Adaptation mechanisms
Deployment-time adaptation
actions
Design artifacts

Figure 3 - life-cycle for adaptable SBAs (Bucchiarone et al., 2010).

The initial phase of the life cycle is *the (Early) Requirement Engineering and Design* in which the adaptation and monitoring requirements are elicited on the basis of the context dimensions that are considered relevant for the considered SBA. During SBA construction, the corresponding monitors and the adaptation mechanisms are realized and refined until the Deployment phase. When the system is running (*operation and management phase*), continuous monitoring is executed supporting the detection of relevant context and system changes. Here, we can proceed in two different directions: executing *evolution* or *adaptation* of the SBA. In the first case we restart the evolution cycle with the requirements engineering and design phase, while in the second case we proceed identifying adaption needs that can be triggered from monitored events and adaptation requirements.

An adaptation need can be formally defined as the identification of the specific problem-situation that demands for adaptation. For each adaptation need it is possible to define a set of suitable strategies that define the possible ways to achieve the adaptation requirements. On the basis of the current situation, the knowledge obtained from previous adaptations and executions, and the available adaptation mechanisms, a reasoner selects the most suitable adaptation strategy that is automatically or manually performed in the *enactment* phase.

Summarizing, the context-aware adaptation requires the definition of a context model together with the definition of the respective adaptation conditions. In addition, the monitoring phase has a crucial role since all the mechanisms able to capture context changes are performed and are responsible to trigger adaptation. Here, it is fundamental that changes are detected as soon as possible to promptly adapt the application to the new conditions. Finally, once that the need of adaptation is detected, there is need for adaptation mechanisms and for modules that promptly select and enact them.

A FORMALIZATION OF CONTEXT AT DIFFERENT LEVELS OF GRANULARITY

In this section we present our context model. It enables the possibility to model various dimensions. Examples are:

- *User*: the person that accesses the application. It is possible to consider different types of users with different roles (e.g., simple user, administrator) that may interact with the application in different ways. Thus, this context dimension contains the information about the privileges, the roles, or the preferences the user has in the application. Moreover, such dimension permits to express the users' goals.
- *Environment*: it can be related to the space and time factors. The space factor may be expressed in terms of an address or may be referred to the environmental condition of the user (the value of some measurable physical parameter). The time factor refers to the information about the time in which the access to the application occurs; it could be expressed in absolute terms (defining a precise date) or it could indicate a part of the day (morning, afternoon, evening, night) or an interval in general.
- *Application:* this context dimension refers to the characterization of the service invoked for the SBA execution. It contains information about all the services together with their status (if a previous execution reported an error, or if it is available), the time of the last failure, and the list of similar services. The latter information could be exploited if a service needs to be substituted with another one.
- *Business context:* it considers the characteristics of the domain for which the application has been developed. The business context usually defines a set of constraints and policies for the application execution.
- *Device:* it specifies the software and/or hardware characteristics that are available at the end user side. Such elements permit to specify, for example, the physical characteristics of the device, the operating system, or the web browser the user is using for accessing the application services.

The level of abstraction to be used to represent the context relevant to an application depends on the application itself and can even vary from time to time. For instance, consider a service offering touristic information about various areas in Europe. In the case the user is located outside Europe, the application may propose, as a default, general information concerning the entire continent. In the case the user is located in Europe, then his/her exact location would be of interest for the application that would propose information about events near to that location. Indeed, another service calculating the taxes the user has to pay on a certain product purchased through some web portal is, at the same time, interested in knowing only the country in Europe in which the user is located. We explicitly represent these different levels of abstraction as part of the context model. Therefore, considering that the context is composed of different dimensions and that each dimension can be represented at different levels of abstraction, which we call *granularity levels*, we provide the following definition.

Definition (Context Model and Context Dimensions): A context model is composed of a set of dimensions $C = \{C_1, \ldots, C_m\}$. Each context dimension can be *elementary* or *composite*.
An elementary context dimension is defined as $C_k = <n_k, V_k>$, where n_k is the name of the context dimension and V_k is the set of its possible values. This set either *categorical* or *interval*. In the former case, the values will be included in a specific vector $V = (v_{k1} \ldots v_{kh})$ while, in the latter case V will be defined by its extremes, i.e., $V = [v_k^{min}, v_k^{max}]$. The actual values of the quality dimensions can be obtained at execution time.
A composite context dimension is defined as $C_k = <n_k, SC_k>$, where n_k is, again, its name and SC_k is a set that includes information about all other context dimensions related to C_k through one of the following two relationships:

- $\mathscr{A}$, which includes all pairs $< C_k, C_i>$ where C_i represents a specific aspect of C_k (think, for instance at the many possible aspects concerning the user context dimension); and
- $\mathscr{F}$, which includes all pairs $< C_k, C_j>$ where C_j represents a finer grained dimension with respect to C_k (for instance, C_k could represent geographical continents C_j could represent a country which is located in a continent).

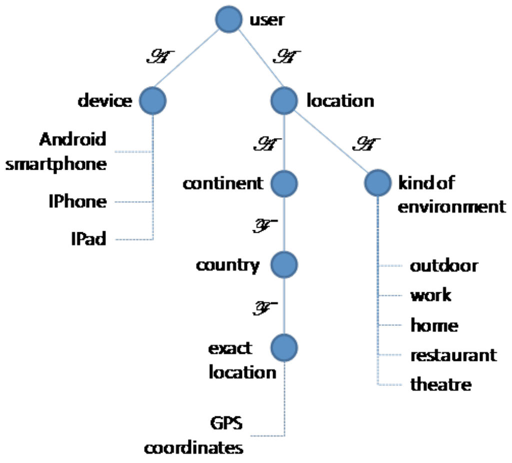


Figure 4. Example of context model.

More precisely: $SC_k = \{<C^k_1, R^k_1>, \ldots, <C^k_p, R^k_p>\}$, where C^k_i (for $1 \leq i \leq p$) are context dimensions and R^k_p are either some $\mathscr{A}$ or some $\mathscr{F}$.

The example in Figure 4 defines the *user* context dimension as characterized by aspects *device* and *location*. *device* is an elementary dimension defined by a categorical set of values. *location* is, in turn, a composite dimension characterized by aspects kind of environment and continent. This last one, is refined into the dimensions *country* and then exact *location*. Specific values associated to the elementary dimensions are highlighted in the figure and connected to the corresponding dimension through dotted lines. The reader should notice that when a pair of context dimensions is in the relationship $\mathscr{F}$, this implies that there exist a way to compute the value of the lower grained dimension starting from the value of the finer grained one. In the example, the GPS coordinates characterizing the *exact location* dimension are used to identify the values for *country* and *continent*.

CONTEXT INSTANTIATION

A context model can be suitable for execution if the transitive closure of each context dimension leads to a set of values. Intuitively, if we refer to the graphical representation of a context model, this means that all its leaves have to be associated to concrete values. Of course, the values associated to dimensions can change over time during the execution of the SBA. In case of discrete and finite sets of values, we can define a State Transition System where the states correspond to possible configurations of a context dimension and transitions stand for possible context dimension evolutions. For each context, a set of dimensions and their level of definition is associated, denoting admissible values. In case of continuous values, we can either define a function that leads from a value to the other, or group values in ranges that are then used in a state diagram. Figure 5 shows an example of state transition system associated with the “kind of environment” context dimension.

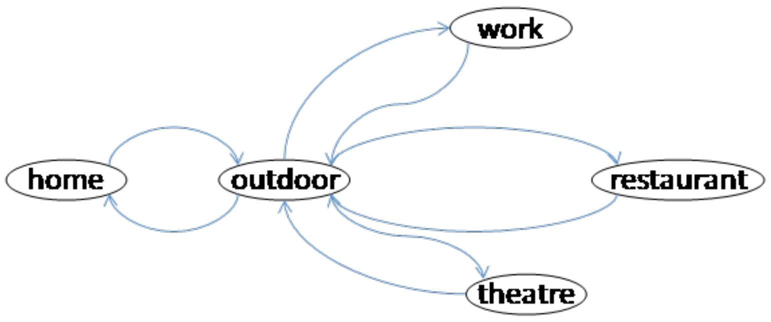


Figure 5. State Transition System for "kind of environment" context dimension

Usually, these relationships between possible values of context dimensions are not captured as they are not under the direct control of the context-aware application. However, describing them explicitly can be beneficial for self-adaptation as it enables the definition *proactive* adaptation mechanisms that, based on an analysis of the state transition system, execute adaptation actions *before* certain context values are actualized. Moreover, it is important to note that each context dimension may evolve as an effect of service invocations, which corresponds to the “normal” behavior of the domain, but also as a result of volatile – “exogenous” – changes of the environment.
One of the aspects to be considered in the design of context-aware service-based applications is the definition of the possible ways to compute the context. We have identified three possible cases:

- A value V_k is *monitored:* data are collected by means of sensors that are characterized by precision and sensing frequency, which can vary during the execution*;*
- A value V_k is *explicitly inserted by the user*;
- A value V_k is *derived:* data are obtained by applying specific assessment algorithms by using estimated or real data. Values associated to a low grained dimension belong to this category.

The collection of context information itself may vary depending on the specific context of an application. For instance, collecting location-base information indoors and outdoors requires possibly different types of sensors or can be inferred by other parameters, such as actions being performed by the user. Therefore context-aware applications with variable context should include in the design the appropriate context capturing and change modules to be able to adapt themselves to different context types dynamically and automatically. In our approach, we assume that the specification of these modules is part of the process design and that the corresponding context adaptation functionalities are realized as part of the adaptation mechanisms used at execution time (see the next section).

CONTEXT-AWARE ADAPTATION OF SERVICE-BASED APPLICATIONS

Adaptation of a SBA is performed by *Adaptation Mechanisms* that are constructed in the *Construction* phase of the life cycle and that can be executed during the *Adaptation Process* that is triggered by *Monitored Events* or by any other external stimulus that can be acquired by the system and that leads to the modification of the SBA.
For this reason a service provider or a service integrator, who compose a set of services, have to specify for each service the dimensions of the context to which it is related. To do this we use the following function:

Definition (Context Mapping Function): At design time (i.e., in the *Requirements Engineering* and *Design Phase*), it is important to define, for each service included in the SBA, the dimensions of the context that can affect its behavior. It is called *context mapping* and can be defined as a function m_z: $s_z \rightarrow C_z$ that specifies, for each service, the set of relevant context dimensions $C_z \subseteq C$. Considering the union of all the sets of relevant dimensions $\cup_z C_z \subseteq C$ we obtain the set of context dimensions associated with the specific application SBA. For each context dimension, it is also necessary to define the sub-context dimension considered that is the particular aspect that will be considered or the granularity with which the context dimensions has to be measured. The sub-context dimensions can vary dynamically during process execution, as well as their granularity. In case, adaptation of context dimensions is also part of the adaptation mechanisms, and the designers must specify, in addition to the set of context dimensions and their sub-context, also the possible changes in the context representation that can be applied during execution, in terms of conditions for context representation variability and context changes (for instance, moving to a finer granularity when a context dimension becomes more critical). It may also happen that the same dimension has to be assessed with different *monitoring frequency* on the basis of context changes. This means that in our approach also the monitoring procedure is adaptive. In fact, for example, in case of emergency, some changes may occur more frequently that in normal conditions and in such situations it is necessary to monitor data more often. On the contrary, it may also happen that the monitoring system acquires more data than needed wasting uselessly resources, so the monitoring rate for the dimensions of interest may be reduced.
As stated above, the context includes users and execution properties. Users' characteristics and preferences can be obtained explicitly, for instance, by filling a user profile, or derived implicitly by profiling users at run-time. Other information such as the users' geographical position, the temporal details, and the actions that characterize the interaction of the users with the surrounding space can be obtained through monitoring. Note that defining the context mapping, we define the requirements for the *monitoring mechanisms* that are tools (that have to be realized in the *Construction phase*) able to gather the needed context data. Since the context mapping specifies the relevant context dimensions and sub-context dimensions, it provides all the information needed for *designing monitoring* mechanisms able to gather data related to a context dimension with different granularities.

As mentioned above, it is also necessary to specify the *monitoring frequency,* that is the frequency with which the context values have to be assessed. The definition of the monitoring frequency is a critical issue since it affects the precision and effectiveness of the adaptation operations. The monitoring mechanisms should be designed in order to be able to self-define the more suitable monitoring frequency, in relation with the frequency with which context changes occur. This adaptive behavior is enabled from reasoning on historical data or on the characteristics of the reference scenario. We assume that we can insert in the process description, in critical points of the process or as a parallel subprocess, a *monitoring dimensions assessment block* which enables the re-evaluation of the monitoring frequency and granularity for context dimensions, that can be based on historical data and provide reactive or proactive evaluations of the context. Reactive evaluations are mainly based on thresholds associated to the dimensions, while proactive monitoring can be induced using learning mechanisms on a training set of process executions.
Once the context mapping is complete, it is necessary to properly capture and define the adaptation aspects (i.e., all the activities and artifacts in the adaptation cycle). In particular, it is important to define when the contextual changes are critical for the SBA functioning (i.e., adaptation triggers) and what should be done or achieved when these changes occur (adaptation needs).

For each considered SBAs, it is also necessary to consider the available or desired *adaptation mechanisms* that are tools that implement a way to execute adaptation. Generally adaptation requires some temporary modification permitting to respond to changes in the requirements and/or in the application context or to faulty situations. Examples of adaptation mechanisms are mechanisms that enable re-composition, service substitution, dynamic binding, compensation, re-execution, evolution, failure.

What we want to deal here are situations where the running SBA need to be resumed from the execution failure (i.e., a service is not available, the location of the user is changed and the WiFi connection is not available, etc...). This means to resume its execution from the point where it has been blocked. Let us consider a SBA that orchestrates a set of services *S*, and operates in the context represented by different context dimensions C_K. When some of these services cannot be executed due to the value of some context dimensions, the execution cannot proceed and the adaptation of the SBA is required. In order to adapt the SBA we need to generate an adaptation process which is the orchestration of services that implements the adaptation strategy. The goal of resuming the execution of the SBA can be expressed as the reachability of a configuration of the context in which the execution of the next application task is possible.

Definition (Context-Aware Adaptation Trigger): When a change in the values associated with some C_k occurs, the application could need adaptation. An adaptation trigger At_k is a predicate composed of an elementary dimension or of a lower grained dimension $C_{k,}$ a comparison operator (e.g., =, >, <) or a set operator (e.g., $\in$, $\notin$) and a value (or set of values) $\subseteq V_k$.

Definition (Adaptation Strategy): an adaptation strategy *as* is a tuple (at, ap, p) where *at* denotes the adaptation trigger the strategy is devised for, *ap* is the adaptation plan, and *p* is the priority associated to the strategy. The last element is important when there is more than one strategy for each adaptation trigger.

The adaptation plan *ap* associated with an adaptation strategy *as* defines the set of adaption mechanisms to use in order to react appropriately on an adaptation trigger.

In order to handle the phases described in the previous section , in Figure 6 we propose a general framework able to manage adaptation in service-based application on the basis of context changes. An adaptable SBA not only is usually able to satisfy some requirements, but it also poses new requirements in terms of monitoring and adaptation aspects. Monitoring requirements concern the need for detecting (part of) those situations that may trigger the need for adapting an SBA. From these requirements, designers should derive the properties to be monitored. These are then observed at runtime by a *Monitoring Engine* that, based on their values, is able to emit some *Monitored Events*. *Adaptation Requirements* are fulfilled by *Adaptation Strategies* that can be executed during the *Adaptation Process* that is triggered by *Monitored Events* or by any other external stimulus that can be acquired by the system and that leads to the modification of the Adaptable SBA.

The architecture we propose is based on the presence of a Context Manager able to gather all the context information needed by the application, generate monitoring events (threshold violations), and assess context changes. Each time that context changes or monitoring events are generated, they are sent to the Event Bus that is responsible for the identification of the events that require the adaptation in the service-based application. Obviously, not all the changes enact adaptation, but only a subset of them. If adaptation is required, information is transmitted to the Adaptation Manager that has to define the most suitable adaptation strategy (i.e., service substitution, re-execution, re-negotiation, re-composition, Compensation, etc…) to activate for the current context. It is worth noticing that an adaptation strategy can realized by means of different adaptation mechanisms (i.e., provided by the adaptable SBA).

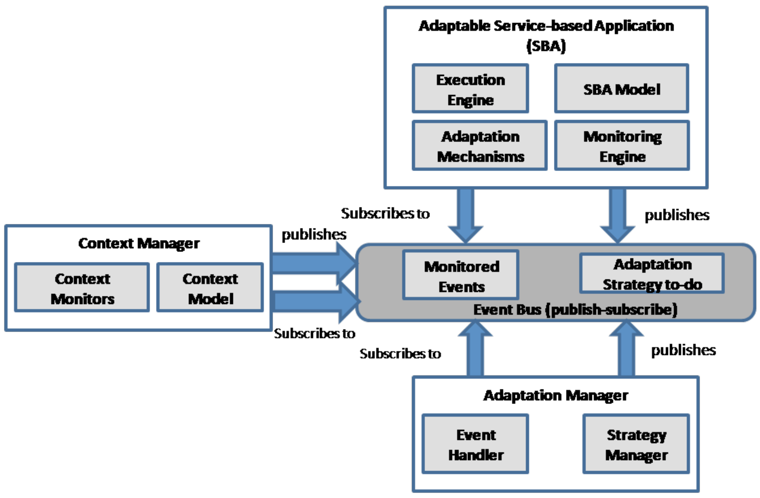


Figure 6. A Framework for Context-aware Adaptation of SBAs

The *Context Manager* is the module delegated to the instantiation of the *Context model* for the related application and to the continuous monitoring of the context variables that contain information about the user, the application, and the environment in which application is running.

The *Adaptation Manager (AM)* reacts to monitoring events arriving from the SBA and from the Context Manager. Based on the incoming events, the AM infers an adaptation trigger that characterizes the cause of adaptation. The component that is in charge to generate the adaptation triggers is called *Event Handler*. The AM acts upon a set of adaptation strategies, each of which defines a process for specific adaptation triggers. The adaptation mechanisms invoked by each adaptation plan are published by the SBA to the Event Bus of the framework and the AM uses them by its subscription to that bus. For example, if the AM selects a strategy with the associated plan (am_i, am_j, am_k), it means that the mechanisms are contacted in that precise order.

AN EXAMPLE OF CONTEXT-AWARE ADAPTATION

In this section a simple real-world example is used in order to better explain our approach and show its validity and effectiveness. Stock management aims at minimizing costs by reducing the amount of stocks in the warehouse but in the meantime guaranteeing the continuity of the production activities. The stocks are defined as the goods available in a certain time instant in a company and that can be used in the production activities or that are ready to be delivered to the customers. A good stock management has to address the following issues:

- the definition of the optimal quantity order to order and store in the warehouse;
- the definition of the period in which a supply is needed

The former issue implies the minimization of the costs related to the stock management that are: supply costs, stoppage costs, stock-out costs and stock-over costs. If the company work in a market in which there is absence of uncertainty, the quantity to order is calculated with the Wilson Model (Hax et al., 1984) for which the stock management issue is related to the supply and stoppage costs and the quantity to order is calculated considering a trade-off between the two costs. On the basis of this method, every time that the stock level decreases under the reorder point, the economic order quantity has to be ordered. Summarizing, the Wilson model is applicable when the reorder point and the economic order quantity have been defined.

The latter problem refers to the identification of the time instant in which it is suitable to place the supply order. In order to address this issue, the reorder point (RP) is determined in a way that it contains goods sufficient to maintain active the production during the supplying phase:

$$RP = PT * cr$$

where PT is the supplying time and cr is the consumption rate.

The problem is that neither a perfect, reliable, and secure supplying system nor an exact method to evaluate the demand exists. Goods may be delivered earlier or later than the expected date. In order to address this uncertainty, it is necessary to consider the safety stock that is the stock quantity than an organization should have in order to face variations over time of the demand and of the procurement time. Considering the safety stock (SS), the formula for the definition of the reorder point is:

$$RP = PT * cr + SS$$

Let us consider the scenario in which an organization has different shops located in different places. Each shop has a local warehouse that receives the goods from a central warehouse each time the stock level is below the reorder point. The central warehouse has always the adequate level of goods to supply all the different shops if the orders request always the same amount of goods and are periodic where the time period between two consecutive orders remains fixed.
This strategy is not optimal since it does not deal with uncertainty. In the shops, it may happen that the demand varies and consequently the shop has to order different amount of goods or the frequency with which the orders are places varies over time. For this reason, the central warehouse has to analyze the requests of the different shops in order to prevent stock-out and stock-over situation. The central warehouse is in charge to perform the following activities: (i) calculate the demand forecast and (ii) select suitable suppliers.
The first activity is a suitable example to describe the importance of the monitoring granularity, proactive adaptation, and dynamic service invocation. The second activity is instead useful to show that some processes can be executed in different ways on the basis of the granularity of input parameters, reactive adaptation, and the use of dynamic service invocation.

DEMAND FORECASTING

Performing a correct demand forecasting is the fundamental input for planning and coordinating the main activities of an enterprise. In fact, demand forecasting is the basis to define the production plan, the procurement strategy and the marketing strategies and to plan the distribution. As a basis for adapting the reorder activities in this process, some characteristics of context parameters themselves may vary in time during the execution of the process for a improving the adaptability of the process. For instance, in the given process two relevant dimensions are associated to the context and its monitoring:

- monitoring frequency T and
- reorder level K.

The identification of these variables is also related to some other context parameters such as: atmospheric condition, number of holiday days in the considered period, season and geographical position (see Figure 7 for the context model). Let us assume that the organization has a database that contains historical data about observations of the demand trend (i.e., descending, ascending, constant) in specific days together with the context parameters that characterize the observed local warehouse. The dataset is stored as a set of records of contextual attributes (e.g., sun, 3.1, summer, sea, city) labeled with the assessed demand trend. In this application, demand forecasting supports stock management in the definition of the most suitable quantity to order and the frequency with which the different warehouses have to be monitored. A proactive monitoring adaptation can be obtained by reasoning on the context of the process to vary process context parameters by means of a classification algorithm and enables the classification of the current situation of the local warehouses in order to obtain an estimation of the demand trend.

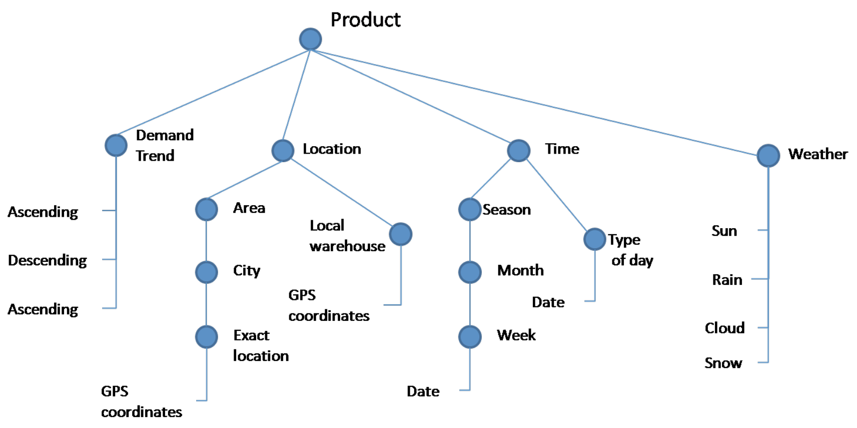


Figure 7. Context Model of the Scenario.

Through the analysis of these data, the responsible of the central warehouse is able to update the quantity of goods to order. The analysis of the demand trend is also used to update the monitoring frequency in order to deal with unforeseen situations. For instance, in Figure 8, as part of the process design a dimensions assessment block is included, to change dynamically the monitoring frequency and the reorder level according to the orders trend.

This part of the process to analyze the demand trend and update both the quantity of goods to order and the monitoring frequency can be executed itself by means of an adaptive service-based application. In fact, this application can be designed as a process in which two services are sequentially invoked (see Figure 8): (i) a service that by means of machine learning techniques and classification algorithms is able to estimate the demand trend (ii) a service that, on the basis of this predicted trend, updates the warehouse data (quantity of goods to order and the monitoring frequency). The latter service will be executed by using dynamic binding and thus, at run-time, on the basis of the considered demand trend the suitable update service will be selected (see Figure 8). For example, if we have a descending demand trend, the service *UpdateStoreDescending* decreases both the monitoring frequency and the quantity to order while if the same values will be increased if we have an ascending demand trend. These changes in the dimensions are performed on the basis of the granularity and sub-context dimensions defined for this process.

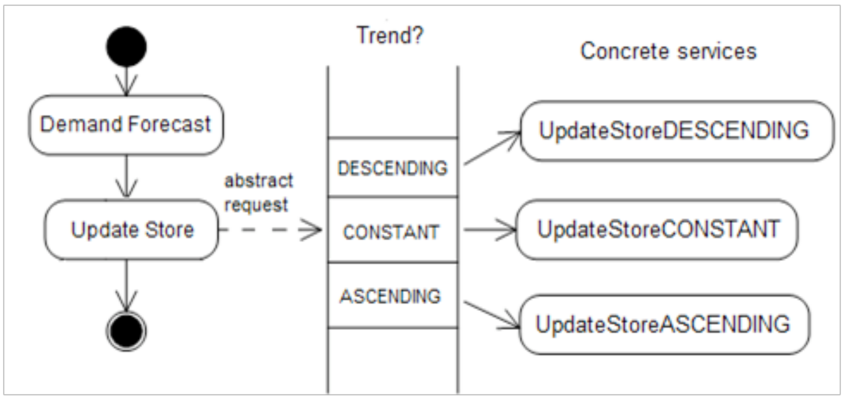


Figure 8 - Demand forecasting dimensions assessment block

SUPPLIER SELECTION

The central warehouse periodically selects the suppliers. For each product, the quantity needed is identified and the most suitable supplier has to be selected. In the stock management, many times the products are classified by considering their "organizational importance". This classification is often performed by using the ABC analysis that supports the categorization of the products into groups. These groups are often marked A, B, and C. Products can be ranked by considering their importance in terms of sales revenues. ABC analysis is frequently combined with Pareto analysis and consequently we can assign to the A group all the products that are related to the 80% of sales, to the B group contains all the products that allow to go from 80% to 90% of sales and the C group contains all the products to go complete the 100% of sales. On the basis of this classification, the products in the A group are particularly relevant since they are valuable and often requested. The products in the B and in C group are less critical. For each product, it is necessary to evaluate the quantity to order and to identify the profile of a suitable supplier on the basis of the importance of the product that is modeled as a context parameter. The supplier analysis will be more or less accurate depending on the importance of the considered product. Therefore, we can model the supplier selection as an adaptive service-based application that uses a *SupplierAnalysis* service. The supplier analysis should be more or less exhaustive on the basis of the product relevance. Therefore, at run time, we need to adopt an adaptation mechanism for the dynamic selection of the most suitable analysis service (see Figure 9).

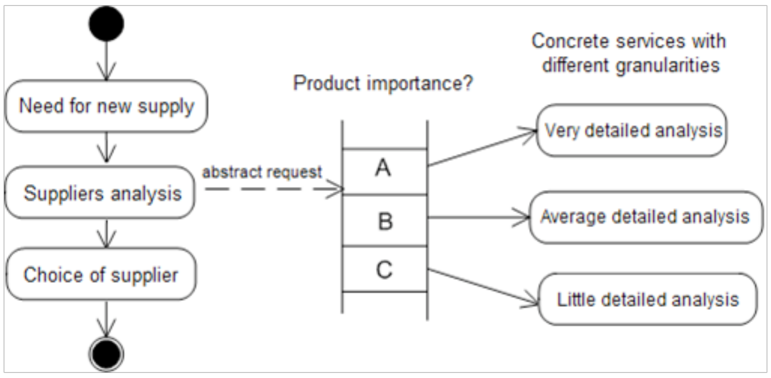


Figure 9 – Supplier Analysis.

IMPLEMENTATION

The adaptable SBA described in the previous section and the context management and adaptation mechanisms have been realized using the Eclipse Helios development tool, the WTP (Web Tools Platform Project) drivers and Axis2 v 1.5.4 functionalities to realize dynamic web service invocation with the open source ESB (Enterprise Service Bus) v 3.0.1, based on Apache Synapse provided by WSO2 consortium. Each abstract service is configured as a proxy service, filtering invocation requests on the basis of the context and service selection rules and invoking the corresponding concrete services. The input (and similarly the output) message, when necessary, is transformed in order to make it compatible with the invoked concrete service. For instance, if data are used with different granularity levels according to the context, the data with the correct level of granularity is send in the invocation of the concrete service. Similarly, sub-context dimensions may be varied in order to make the service invocation compatible with the requested input for the service, based on the context model. The class mediator class, realized in Java in Eclipse synapse-core1-1-1.jar library and inserted in the ESB, decides at run time on the basis of context parameters which concrete service to invoke.
The services have been realized as Java classes using Apache Tomcat 7.0.11 and the MySql 5.1.49 DBMS. For each adaptive service, an abstract service has been created, and the relevant concrete services are dynamically invoked on the basis of the context (e.g., in the example of Figure 9, the abstract service supplier analysis is inserted in the process, and in case of context *ProductImportance = A*, the concrete service *VeryDetailedAnalysis* is invoked.

For process design, the Eclipse Plug-in BPEL designer the Apache ODE (Orchestration Director Engine) 1.3.5 have been used. For the Context Management modules, Context type changes are obtained with ad-hoc developed services, such as the the UpdateStore service, invoking the different UpdateStore concrete services corresponding to a given trend evaluation. The Weka toolkit (Weka 3.6.4, realized at University of Waikato [10]) has been used as a basis for proactive adaptation, to forecast context trends in context changes, using the NBtree classification algorithm. The updates in context models are designed as separate processes which update the context data model.

CONCLUDING REMARKS

In this paper we have presented our approach to build context-aware self-adaptable SBAs. The approach features a rich context model that put together two interesting aspects: a) the fact that contextual information can be available at different levels of granularity and that such granularity can vary over time during the execution of the application and b) the fact that evolution of context values are explicitly described in the model either as state transition systems or as continuous functions.
The first aspect allows the SBA to be designed by exploiting the possibility to change level of granularity for collecting contextual information thus offering to the users more precise tailoring to their needs. The second aspect enables the possibility to define adaptation plans in a proactive way thus anticipating the needs for adaptation that may arise during the application execution.
While the example presented in this paper represents a first experience of usage of our framework, a deeper evaluation is planned in the short term. This evaluation will concern the extensive experimentation of our context model both in terms of its expressive capabilities and of its ability to trigger proper adaptation actions in an effective and timely way.

ACKNOWLEDGEMENTS

Research leading to these results has received funding from the European Community's 7th Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

REFERENCES

Ardagna D., Pernici B. (2007). Adaptive Service Composition in Flexible Processes. IEEE Trans. Software Eng. 33(6), (pp. 369-384).

Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: PAWS: A framework for executing adaptive web-service processes. IEEE Software 24(6), 39–46 (2007).

Autili M., Berardinelli L., Cortellessa V., Di Marco A., Di Ruscio D., Inverardi P., Tivoli M. (2007). A Development Process for Self-adapting Service Oriented Applications. ICSOC 2007 (pp.442-448).

M. Baldauf, S. Dustdar, F. Rosenberg (2007). A Survey On Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4), p. 263-277, Inderscience Publishers.

Baresi, L., Guinea, S., Pasquale, L.: Self-healing BPEL processes with Dynamo and the JBoss rule engine. In: Proc. of ESSPE 2007, pp. 11–20. ACM Press, New York (2007).

Cristiana Bolchini, Carlo Curino, Elisa Quintarelli, Fabio A. Schreiber, Letizia Tanca: Context information for knowledge reshaping. Int. J. Web Eng. Technol. 5(1): 88-103 (2009)

Bucchiarone A., Cappiello C., Di Nitto E., Kazhamiakin R., Mazza V., Pistore M.(2009). Design for Adaptation of Service-Based Applications: Main Issues and Requirements. Proc. of ICSOC/ServiceWave Workshops (pp. 467-476).

Bucchiarone A., Kazhamiakin R., Cappiello C., Di Nitto E., and Mazza V. (2010). A context-driven adaptation process for service-based applications. In Proc. of PESOS 2010.

Romolo Camplani, Fabio A. Schreiber, Letizia Tanca, Diego Viganò: Towards autonomic pervasive systems: the PerLa context language, Electronic Proceedings of the 6th International Workshop on Networking Meets Databases (Co-located with SIGMOD 2011), pp. 1 - 7, Athens, June 12-16, 2011.

Cappiello C., Comuzzi M., Mussi E., Pernici B. (2006). Context Management for Adaptive Information Systems. Electr. Notes Theor. Comput. Sci. 146(1), (pp. 69-84).

Chaari T., Laforest F., Celentano A.(2007). Adaptation in context-aware pervasive information systems: the SECAS project. Int. J. Pervasive Computing and Communications 3(4) (pp. 400-425).

Colombo, M., Di Nitto, E., Mauri, M.: SCENE: A service composition execution environment supporting dynamic changes disciplined through rules. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 191–202. Springer, Heidelberg (2006).

Dey A. K. (2001). Understanding and Using Context. Personal and Ubiquitous Computing 5(1), (pp. 4-7).

Fahy, P. and Clarke, S. (2004) 'CASS – a middleware for mobile context-aware applications', Workshop on Context Awareness, MobiSys 2004.

Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process life cycle. In: Proc. of ICEIS, vol. (3-2), pp. 154–161 (2008).

Hax, AC. and Candea, D. (1984), Production and Operations Management, Prentice-Hall, Englewood Cliffs, NJ, (pp. 135).

Higel S., Lewis D., Wade V.P. (2005). Realising Personalised Web Service Composition Through Adaptive Replanning. OTM Workshops (pp. 49-58).

R. Hirschfeld, P. Costanza, O. Nierstrasz . Context-oriented programming. Journal of Object Technology, 2008

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G. and Altmann, J. (2002) 'Context-awareness on mobile devices – the hydrogen approach', Proceedings of the 36th Annual Hawaii International Conference on System Sciences, pp.292–302.

W. Hummer, Ph. Leitner, A. Michlmayr, F. Rosenberg, S. Dustdar: "VRESCo - Vienna Runtime Environment for Service-oriented Computing"; in: "Service Engineering: European Research Results", S. Dustdar, F. Li (ed.); Springer, 2011, (invited), ISBN: 978-3-7091-0414-9, 299 - 324.

Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.P.: Extending BPEL for run time adaptability. In: Proc. of EDOC 2005, pp. 15–26. IEEE Press, Los Alamitos (2005).

Kongdenfha, W., Saint-Paul, R., Benatallah, B., Casati, F.: An aspect-oriented framework for service adaptation. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 15–26. Springer, Heidelberg (2006).

Martín E., Carro R. M., & Rodríguez P. (2006). "A mechanism to support context-based adaptation in m-learning", in EC-TEL (pp. 302–315).

Narendra, N.C., Ponnalagu, K., Krishnamurthy, J., Ramkumar, R.: Run-time adaptation of non-functional properties of composite web services using aspect oriented programming. In: Kramer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 546–557. Springer, Heidelberg (2007).

Wenjia Niu, Gang Li, Zhijun Zhao, Hui Tang, Zhongzhi Shi. Multi-granularity context model for dynamic Web service composition. J. Network and Computer Applications, 2011: 312-326

Sandonini A., Adattivita' in applicazioni orientate ai servizi basata su un'analisi proattiva e granulare dei dati contestuali, Master thesis, Politecnico di Milano, July 2011

Spanoudakis, G., Zisman, A., Kozlenkov, A.: A service discovery framework for service centric systems. In: Proc. of SCC 2005, pp. 251–259. IEEE Press, Los Alamitos (2005)

Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The meteor-s approach for configuring and executing dynamic web processes. Technical report, University of Georgia, Athens (2005).

Appendix E: Using the Cloud to Facilitate Global Software Development Challenges

Sajid Ibrahim Hashmi, Viktor Clerc, Maryam Razavian, Christina Manteli, Damian Andrew Tamburri, Patricia Lago, Elisabetta Di Nitto, and Ita Richardson

Abstract— With the expansion of national markets beyond geographical limits, success of any business often depends on using software for competitive advantage. Furthermore, as technological boundaries are expanding, projects distributed across different geographical locations have become a norm for the software solution providers. Nevertheless, when implementing Global Software Development (GSD), organizations continue to face challenges in adhering to the development life cycle. The advent of the internet has supported GSD by bringing new concepts and opportunities resulting in benefits such as scalability, flexibility, independence, reduced cost, resource pools, and usage tracking. It has also caused the emergence of new challenges in the way software is being delivered to stakeholders. Application software and data on the cloud is accessed through services which follow SOA (Service Oriented Architecture) principles. In this paper, we present the challenges encountered in globally dispersed software projects. Based on goals mutually shared between GSD and the cloud computing paradigm, we propose to exploit cloud computing characteristics and privileges both as a product and as a process to improve GSD.

I. INTRODUCTION

Advances in technology and communication channels has had a positive impact on business growth as the exchange of information has become more timely, accurate and available. Because of this, business organizations are no longer reluctant to outsource software development and to have development operations in multiple geographical locations. They strive to make use of customized business models to maximize their benefits. In addition, from the marketing perspective, the goals of globally sourced development [10] include making use of international physical and material resources, reducing time to market, and taking advantage of marketing business opportunities.

In the remainder of this introduction section, we highlight the context of this research, the research question, the objective of the research, and the research methodology. Also, we present a synopsis of the cloud computing, challenges faced by GSD, and our motive for using the cloud paradigm to support GSD.

A. Context

In the global environment, outsourcing software development projects to low cost economies is becoming increasingly popular, especially as there is the expectation that companies who embark on GSD strategies will gain and maintain economic advantage through numerous technical and commercial factors [1][2]. This increase in GSD implementation is supported by the availability and accessibility of communication tools as they enhance the options to use a remotely located workforce [3]. The business models in low cost countries have provided capable and willing workers who undertake outsourced and offshore software development [4]. This in turn provides cost reduction in software development projects [5]. However, outsourcing software development to organizations at various outsourcing destinations is not an easy and straightforward task [8][9][10][11] and organizations very often face difficulties due to global distance and the involvement of the development teams which are geographically distributed.

B. Research Question

GSD is software development incorporating teams spread across the globe in different locations, countries, and even continents. We are motivated by the fact that conducting software projects in multiple geographical locations is likely to result in benefits such as cost reduction and reduced time-to-market [14][19], access to a larger skill pool, proximity to customer, and twenty-four hour development by following the sun [60]. But, at the same time, GSD brings challenges to distributed software development activities due to geographic, cultural, linguistic, and temporal distance between the project development teams.

In order to meet the different challenges posed by GSD, we suggest making use of the cloud computing paradigm and illustrate that it has potential to enhance the usefulness of GSD. We argue that different types of geographic and cultural issues can be addressed by making use of different cloud computing realizations such as PaaS (Platform as a Service), IaaS (Infrastructure as a Service), and SaaS (Software as a Service). Since data in the cloud is accessed through services [38], we study its characteristics in the light of Service- Oriented Architecture (SOA). Furthermore, we argue that the cloud can facilitate GSD both as a process and as a product. The former one could have implications for the GSD business model in which service providers are organizations and services are parts of a GSD process, for example, requirements, design, coding, and testing. SOA as a product is developed, run, and distributed globally. The idea is to identify different types and domains of GSD issues and investigate the potential of the cloud to address those.

C. Objective of the Research

This paper proposes the development of Global Software Development (GSD) using the cloud computing paradigm, based on our understanding of current GSD and SOA methods from literature, and our overall project aim is to propose the re-construction and improvement of the GSD process. This is done through the use of cloud computing and SOA. We discuss how the GSD process can be aligned with SOA, and how GSD products can be implemented using services. We have established that, for example, some web tools such as Wikis support GSD communication processes. However, we question whether these can be streamlined and re-organized by defining how exactly GSD can work better by making use of a service based environment.

Initially, we identify problem areas in GSD and subsequently, propose the support of GSD development activities through services. The emphasis is on facilitating collaboration activities among GSD teams by structuring those activities. Our rationale is that we can parallel the GSD situation with manufacturing supply-chain management where systems used are composed of ready-to use service-oriented systems. The reason services are widely adopted in industry is because they can be integrated seamlessly. This has resulted in benefits to industry such as increased return on investment and reduced information technology costs [5]. We argue that services to support GSD activities could be developed in the form of service based systems and that what we need are heterogeneous services which could support different development activities. Moreover, output from one service could be taken as input to the next, in cases, where those services supported interrelated activities. In this article, terms like SOA (Service Oriented Architecture) and the cloud have been used interchangeably as different representations of the cloud are being accessed using services.

D. Research Methodology

In order to conduct this research, our literature review studied characteristics of services (both SOA and the cloud). We also identified challenges faced by GSD. Following this step, we held a workshop, attended by all of the authors of this paper, each of whom has research and/or industrial expertise in GSD and/or SOA. During this workshop, through interactive discussion and brainstorming, we developed the concepts presented in this paper. To do this, we summarized the GSD challenges and requirements and investigated the potential of SOA based cloud services [47] to address these. We are embarking on further research to understand whether these indeed can be of value to both the industrial and research communities.

E. Cloud Computing

Cloud computing is an internet based computing paradigm in which shared resources like software, hardware, and information are provided to the subscribers on demand [17][18][26]. NIST [55] defines cloud computing as a model for enabling convenient and on demand network access to shared computing resources that can be managed and provided rapidly with minimal effort. The aim is to construct a low cost computing system by using certain entities without compromising on computing capabilities. Depending on the type of shared resources, the cloud paradigm can have different implementations like IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service), to dispense computing capacity to end users.

Infrastructure as a Service (IaaS) includes the delivery of hardware such as processors and storage as a service, e.g., Amazon Elastic Cloud (EC2) and Simple Storage Service (S3). In other words we can say that it delivers a platform utilization environment as a service. Instead of physically purchasing hardware and software infrastructure, clients buy such resources as a fully outsourced service.

In addition to the infrastructure, Platform as a Service (PasS) occurs when a software platform is provided on which systems can be run. This includes the delivery of programming platforms and tools as a service. This kind of cloud computing provides a development environment and the infrastructure provider's equipment can be used to develop programs which are delivered to end users through internet and servers.

Software as a Service (SaaS) occurs when applications are delivered as services using IaaS and PaaS. This implementation of the cloud focuses on separating the ownership and possession of software from its use [18]. It is based on the idea that software functionality could be provided as set of distributed services that could be configured and bound at delivery time, to avoid the current limitations with software use, deployment, and evolution [18]. Since cloud computing stimulates the provision of online services via the World Wide Web, software can be hosted on web servers as services [18]. Thus, the advent of SaaS within the cloud computing paradigm has created new opportunities for organizations to communicate and coordinate among themselves.

F. GSD Challenges

With the emergence of technologies in a world which has become increasingly globalized, the relationship between culture and management of remote work has become an unavoidable issue which needs to be addressed [15]. Because of distance among the software development teams, GSD encounters certain challenges in terms of collaboration [61], communication [62], coordination [63], culture [64], management [65], organizational [66], outsourcing [35][67], development process [68], development teams [16][69], and tools [29][70].

Global distance comprises of four elements: geographic, cultural, linguistic, and temporal distance [57][58]. Geographic distance occurs as the teams are dispersed across countries. Cultural distance occurs due to teams being made up of members from different cultures, and the additional expectation that each member will understand and support each other's culture. When team members speak in different languages, there needs to one chosen language for work purposes, and as this is everyone's first language, linguistic distance occurs. As teams are geographically dispersed, there is the additional difficulty of temporal distance – members working across different time zones [49][50]. Each of these differences individually causes problems within GSD teams, and the culmination of these differences into global distance can and do impede global software development projects [12][13]. Thus, the management of globally outsourced software development has been accepted as a difficult and complex task [14]. These four types of GSD challenges are addressed using the SOA based cloud services (Table 1).

Collaboration Challenges	**Issues**	**Negative Impact on Software Project**	**Facilitating GSD Using Services (SOA/Cloud)**
Geographic	Distance Time Knowledge transfer Tools	Communication gaps Project Delays Ambiguity on technical aspects Unequal quality levels across the software development sites	Dynamic binding, runtime adaptation, and timely availability of required services could help dealing with geographic issues. Also, availability of SaaS could diminish installation overheads at each development location.
Cultural	Unequal distribution of work Lack of Trust, Fear	Increase in cost Poor skill management Reporting problems	Service could maintain a fair distribution of work between the teams. Only a specific person will be responsible for the task assigned to thus skill management would be easier too.

Linguistics	Frequency of communication Knowledge transfer	Loss in project quality Invisibility on project development Ineffective project management	Run time evolution of services can meet with the linguistic issues. Also, isolation of each task and related information as a service can ensure right level of knowledge transfer.
Temporal	Lack of Motivation Less visibility Risk	Loss in project quality Poor management of configuration Chances of project artifact loss	The cloud service models imply that the data resides on a centralized location where inventory of services is maintained. Services maintain a registry where all of them are stored. This attribute could be used to store and retrieve configurations.

Table 1. GSD challenges potentially facilitated by the use of services

G. Motive for Using the Cloud for Supporting GSD

One of the missions [59] of the cloud architecture is to provide services to customers by not only managing them but optimizing them by taking into consideration economies of scale. The cloud model is composed of three service models (IaaS, PaaS, and SaaS), five essential characteristics, and four deployment models [59]. The cloud deployment models - Private, Community, Public, and Hybrid - define the scope of the cloud solution. The cloud model is discussed in terms of creation and provision of services [20] which means that it supports services. Since SOA runs a mechanism for development and management of distributed dynamic systems and it evolved from the distributed component based approach [21], we argue that it has potential to cater the challenges of GSD where a project is developed across different geographical locations. Our thesis is that GSD challenges can be overcome through Service Oriented Architecture (SOA) support. This will contribute increased interoperability, diversification, and business and technology alignment. Moreover, the vision behind this architectural paradigm is to set up common goals and objectives to improve the collective effectiveness of the enterprises participating in globally distributed projects. Since software processes are software too [22], we argue that the cloud has potential to reinforce GSD as a process. Initially, we considered the use of standard procedures to meet the quality challenges posed by GSD. But, since organizations have to interact dynamically in global environments, these standard procedures cannot scale up to support dynamism (which is a main feature of SOA). Moreover, the ideology posed by both SOA and GSD is somehow similar [1, 23], for example, coordination, transaction, context, execution monitoring, and infrastructure. In addition, SOA is one of the main technical foundations of the cloud [51].

For GSD, the use of collaboration tools among teams is not new. Existing research has already proposed further work in this regard [7][24][25]. We adopt the idea of SaaS for GSD to make use of properties of both cloud and SaaS, such as reusability, reliability, extendibility and inexpensiveness [27][28]. Teams with frequent communications among their members are likely to collaborate better. Thus, this frequent communication is important to make full use of GSD advantages, e.g. improved productivity, reduced time to the market, and reduced cost. However, oral communication is prone to confusion and misunderstanding. One way could be to minimize the need for communication but such strategy would emphasize on the involvement of more dedicated personnel from each development site [48] which could not be feasible either. At the same time it is important for the communication media to be formal, flexible, and evolvable to ensure the collaboration mechanisms work effectively.

GSD teams also need to collaborate effectively and the attributes of the cloud paradigm, especially SaaS, can be used to facilitate efficient collaboration between geographically distributed teams during software development phases such as requirements, design, coding, and testing. The characteristics and the architecture of the cloud model itself has the potential to fulfill the GSD task requirements. For example, cloud deployment models allow certain trusted partners (which could be GSD team members) to share resources among themselves. Service models may not only provide access to collaboration and productivity tools but also allow network access to computing resources, and the "use as you go" feature is likely to reduce the overall project costs across multiple development sites as computing resources and infrastructure is not required up- front.

We investigate the impacts of the aforementioned collaboration challenges and suggest the likelihood of using the cloud to address them. We expect to achieve efficiency in collaboration through using the cloud in different implementations. The essence of using this paradigm to facilitate GSD is that instead of acquiring and owning the software and project data, GSD team members can access and subscribe to some of the software at a time (according to the need) in the form of services. In addition, we want to take advantage of the SOA characteristics [43] like loose coupling, service composition and negotiation to facilitate a similar level of development practices across multiple sites. Moreover, the service provider and user are important to the technical and economic changes made possible by cloud computing. In our model, this concept of provider and consumer is similar to the SOA paradigm.

II. GSD AS A SERVICE ON THE CLOUD

In this section, we describe how GSD as a service can facilitate and improve how GSD is carried out. We discuss certain GSD challenges and provide a rationale as to how the cloud service models can address them respectively.

A. GSD Services Concept

Figure 1 illustrates the concept of using the cloud paradigm to support GSD. Service standards and policies are defined by engineering and project management personnel. Different GSD development sites (represented as GSD1, GSD2,...,GSD5) are deployed on a private cloud which covers all geographically distributed development teams.

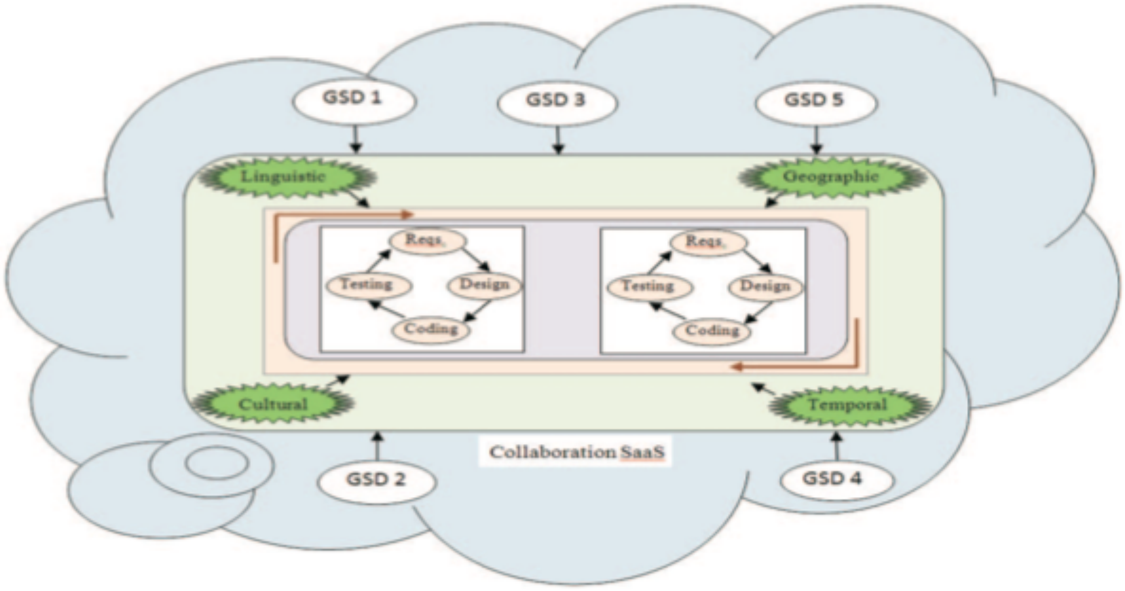


Figure 1. Using the cloud to support GSD product and process activities

We propose that this concept can support the reduction of difficulties caused by global distance. For example, the use of services itself reduces the distance factor to meet geographic and temporal challenges. As far as cultural and linguistic challenges are concerned, the provision of multilingual services based on the location of GSD teams could improve the problem. We consider an example scenario to understand the GSD collaboration challenges that could be minimized using the cloud paradigm. Suppose that an organization in Ireland (GSD1) outsources a software component development to a company in Germany (GSD3).

As part of the project requirement, people in both countries have to communicate to exchange information on different development phases and tasks. The project manager from GSD1 sends on some important instructions regarding requirements and architecture of the potential system. Not only should this information be conveyed to the concerned team member but there should be some assurance that it has indeed reached them, without the risk of being lost or disclosed to other GSD locations. But, the concerned team member is a novice in the language and also needs to have those instructions translated into his local language. Thus, a translation service is required on both sides to facilitate the task.

One would argue that the translation task could be facilitated by a simple word parser, and the use of cloud and services seemed to be irrelevant, but the situation is not as simple as it appears to be. Communications between the teams could involve some other artifacts such as design documents, code snippets, and legal and financial negotiations. All of this could be made available on the cloud in the form of services which could be accessed by the authorized team members. Also, cloud services can evolve with changes in the associated business [18], for example, such a change might occur in the form of a financial or money transfer service after both companies agree on terms and conditions. Using our proposed system would result in the elimination of GSD3 overhead, i.e. storage of project artifacts and information, as everything would be stored on cloud infrastructure and would be accessible from there in the form of services. Table 2 list down the characteristics [46][55] of the Cloud which can be potentially beneficial for GSD.

Virtualization	Courtesy of this privilege, cloud providers can enhance their infrastructure to accommodate in case there is growing demand for services. Usually, a combination of hardware and software are used on the provider side to meet with the scaling requirements.
Reduced Cost	Costs in the cloud do not include server side infrastructure and equipment costs. Moreover, pay as you go model ensures that subscribers are bound to pay for only those resources which they use. In short, the distribution costs of software are reduced.
Scalability	On-demand provision of application software provides scalability, which results in greater efficiency. Whereas cloud based application development platforms provide with high level of scalability thus making the developed application to coup with the fluctuation demands.
Infrastructure	Providers' applications are run on a cloud infrastructure from where a consumer can access those. Similarly, consumer-modified information or application can be deployed on the same infrastructure as well. The privilege is that the consumer does not have to deal with the underlying infrastructure.
Performance	The cloud paradigm can support various levels of performance requirements like service scaling, response time, and availability of the application based on the needs of the consumers. In addition indirect performance measures may also be achieved by eliminating the overheads involved with installation procedures and reduction in unnecessary reduction among the applications running on the cloud.
Multi Tenancy Support	Public clouds are elastic in nature as their consumers are not limited. More importantly, consumers' workloads are isolated to provide privacy. However, the number of consumers can be restricted by opting out a specific deployment model.

Table 2. Supporting characteristics of cloud computing

A. GSD Challenges and Requirements

During our workshop, we identified GSD challenges and requirements which could be, in our view, solved through using cloudF architecture.

1) Coordination

Coordination among distributed teams is important to GSD but geographic distance negatively affects the ability to collaborate [23]. For building complex systems, coordination requires interaction over sequences of operations. However, often, due to collaboration within different time zones, employees have less time to coordinate their work.

As a product, cloud services ensure interactions among different activities. For example, interaction between the service consumer and provider on finding and binding of services is independent of the geographical distance. SOA puts an emphasis on adding transactional guarantees to facilitate the interaction in the coordination framework [31]. For example, standards have been proposed by IBM [32][33] and suggested by Sun [34].

Since cloud computing is the key service delivery platform in the field of service computing [44], *as a process*, it could allow resource sharing not only for infrastructure and application resources, but also for software resources and business processes [45][46]. These advantages are likely to support different disciplines, for example, Infrastructure as a Service (IaaS) could help provide different GSD teams with resources such as computing power or storage provisioning to store project related data. Software resources may consist of middleware and development resources like application systems, database servers, and operating systems. The advantage of using fist two types of resources as a service is that they are never wasted after the project is over - instead, they can be unsubscribed. Application resources could assist in providing SaaS with necessary interfaces that could facilitate collaboration and sharing of information among the teams.

B. Support of Technical Development

A variety of special purpose services can be used for *process* related software development activities e.g. requirements, design, and testing. Services which support different process activities, can be combined together to facilitate the whole process. As shown in Figure 2, supporting development process activities in the form of services can help alleviate geographic and distance challenges.

As a product, shifting the provision model from Software as an Application to SaaS removes the dependencies and challenges in terms of architecture and task dependencies that traditional software development and reuse models impose. Moreover, it can reduce cost by facilitating reuse of services which provide similar operability for software application development. Hence the development is reduced on building similar business applications as the only challenge which remains is the identification of suitable services which can serve the purpose required.

C. Geographical Distance

Physical distance removes the opportunity for face to face communication. *As a product*, the Platform as a Service representation can provide a development platform with set of services to assist application development and hosting on the cloud. It does not require any kind of software downloads and installations [2], and because of its characteristics, has the capacity to support geographically distributed teams. Moreover, the philosophy of the cloud paradigm is to facilitate a pool of shared hardware and software resources.

Facilitating global software development activities *as a process* in the form of services, can overcome many software limitations involving software evolution, reuse, and deployment. Such a model is likely to open not only new opportunities for the business but also the way software is being developed, i.e. services become part of GSD processes being provided by the outsourcing organizations.

D. Global Project Optimization

In GSD, it is important to share the information in terms of work performed by distributed teams. Communication and awareness capabilities should be provided by integrating this information not only into a collaborative environment [1], but also to maintain a rich “project memory” [34].

Provision of this information exchange on software development activities *as a process* is likely to reduce the software installation costs across different development sites. In addition, it can make collaboration more instant and flexible because of the customization and scalability attributes of the cloud.

SOA not only manages service execution and output information, but also keep track of the new information without any changes to the underlying infrastructure [23]. This unique feature can ensure team management and coordination by means of its use *as a product*, by scaling on to the existing project information.

E. Optimizing Globally Distributed Software Development

It is true that geographic distance affects the ability to collaborate [1]. Moreover, it has been reported that communication and collaboration declines as the distance between the two working location increases [28]. *As a process,* cloud based collaboration among GSD teams is likely to diminish the deficiency caused by distance as services are free from geographical boundaries. Yet another type of resource in the cloud could be business process [45], which may facilitate the optimization of the overall technical software development.

As a product, it can serve as an intermediary to facilitate users to access and communicate with the cloud. The services involved in a system can change with the change in the associated business in terms of requirements, and can perform this change dynamically. The reason for change in GSD could be the availability of yet another programming task or a need to collaborate on a task which is already underway.

F Eliminating the Strategic Issues

In GSD, ownership is often lacking [1]. Service ownership is a concept which allows the service users to focus on their core activities; it also helps the service provider with an opportunity to take advantage of economies of scale [56]. *As a product*, well defined ownership exists for each service, and GSD users benefitting from such services can enjoy this privilege. This ensures that services are used in a way to give the most to a business.

As a process, SOA addresses ownership by collocating service provisioning by service development. The wrapping of a GSD task into an independent service can promote ownership, as it can be used exclusively. In this case, the outsourcing company (the provider) could be the one to convey the project requirements or architecture knowledge to a specific GSD team or to a single member without notifying others. Thus, it can incorporate privacy by increasing the eeling of ownership.

G. Enhancing Communication Among Teams

The structure of multi-site software development mimics the team structure [37]. The main distinctive feature of the cloud is that it allows rapid elasticity, making it straightforward for the service provider to dimension the resources necessary to support a service dynamically depending on the service demands [39]. Thus, investigating the potential interactions among the stakeholders would enable getting insight into the service creation process for collaboration among GSD teams. These interactions are likely to be among the outsourcing organizations and the teams jointly working on the same project.

As a product, appropriate service definitions (including their descriptions) may act as proxies for communication and hence may reduce the need for cross site communication. This privilege could be useful when teams from different time zones find it difficult to collaborate. On a technical level, the SOA paradigm provides an appropriate mechanism for cross platform data exchange and sharing by message passing, service search and collaboration [30]. In addition, the SOA and the XML-described data, information and knowledge can combine the different loosely coupled subsystems.

H. Managing Project Knowledge Transfer

The transfer of requirements and architecture knowledge across development sites is an issue in GSD. Services are likely to wrap this knowledge using the correct abstraction level. Using services *as a product* diminishes the need for sharing knowledge as the constituent services have sufficient description about themselves, and because this knowledge is developed locally. Two knowledge transfer issues in GSD which exist are requirements and architectural knowledge. With SOA, a sufficient description and transfer of the requirements knowledge diminishes the need for transfer of other forms of knowledge. Moreover, coordination aspects are hardly needed in services as they are isolated.

In terms of *as a process*, isolating any task as a service helps to identify right abstraction level for the transfer of such information within the task *as a service.* This form of service provision could take care of how requirements should be provided, what the outcome will be, and what should a GSD team member expect from others. Hence it can help managing the knowledge of a distributed project.

I. Execution Monitoring

In GSD, the potential order in which components interact should dictate the decision on the interaction among the corresponding teams [1]. Moreover the use of Application Programming Interfaces (APIs) promotes isolation and reduced information sharing [36]. A crucial aspect is responding to the constant changes in the business requirements. *As a product*, SOA can ensure the correct order of service execution by a central scheduler, which controls the execution of the service and consequently the right order of communication between the project partners in GSD. *As a process*, these services may share their execution context among each other to guarantee the correct execution order. To support the collaboration activities among geographically distributed teams, the concept of execution monitoring can be used as the basis for designing the collaboration process.

J. Eliminating Project and Process Management Issues

In GSD, a centralized configuration management system should be made available to manage project artifacts produced out of components being developed across multiple locations. *As a product*, this task can be facilitated by service registries which serve as databases for services. Potential users can find and bind any service using the service description provided with these registries. Or, alternatively, since services are a black box, their use is likely to eliminate the need for centralized configuration management within their scope.

Cooperation and coordination is required to obtain trust between two or more parties [52]. Lack of trust is always likely to reduce the team cohesion. Teams with higher trust are coordinate better to achieve better performance [53] which could make management an easier task. A goal of cloud computing is that its users must be able to access its different implementations at any time [54].

K. Technical Issues

In GSD, a modular approach for software development has been suggested [1], but dependencies are likely to exist among components in the running version of the software. This nature of such a project would require evolvable software which could cope with the challenge of component and functional dependencies. *As a product*, services are loosely coupled and independent in nature, with minimum dependencies among them. This characteristic can not only minimize the task dependencies but ultimately eliminate the risk factor because, in case of a service failure, the failed component can be replaced with another one at run time. This dynamic replacement is one of the distinguishing features of services.

Testing software is usually the most costly phase in software development and it can be responsible for over 50 percent of development costs [41]. Therefore, this phase often becomes responsible for the ultimate profitability of the product [42]. Carrying out testing activities correctly is important as the quality assurance, financial incentives, and customer satisfaction of the end product often depend on the testing activity [40]. Making use of standard procedures to meet the quality challenges posed by GSD is important, but since organizations have to interact dynamically in global environment, these standard procedures often cannot scale up to support dynamism. On the other hand, dynamism is supported by SOA by means of run time evolution and on demand provision. Facilitating collaboration on testing in the form of the services *as a process* could ensure higher quality levels.

In normal circumstances, different software tools [29][70] are used to facilitate not only GSD development but also collaboration among the teams. Unavailability of tools at the right time or version misalignment can cause delays in global software projects. Use of collaboration tools *as a service* can reduce the overhead of tool installation. Using the SOA paradigm, tools, data, and workspace could be stored and accessed from the cloud, thus eliminating the need for tool installation. The purpose is not only to cater with the version issue, but also provide GSD teams with all tools required for the project.

III. DISCUSSION

In order to propose the use of cloud for GSD, it is important to have a comprehensive understanding of the GSD processes. We do not expect that all will be served by this technological paradigm, but we do believe, if designed correctly, GSD can be successfully supported by services. For example, in a context where different GSD locations are inter-connected and are using the cloud, all of them may not have the same level functional needs. Determining different level of needs for service provision could be one of the major concerns among different GSD locations.

The concept of SaaS itself continues to be subject to evolution and revision. In addition, the availability and subscription of these services because of different types of dependency relationships among cloud users (tenants) could be considered as a challenge for using the cloud to support GSD. Moreover, in terms of project knowledge transfer across global software development sites, the right level of abstraction of the useful codification as well as the reduction in tacit knowledge will remain an issue. Since the main usage in services comes in connecting pieces of information, sharing services across different domains and enterprises is also likely to result in further security issues.

IV. CONCLUSIONS

We have proposed using the cloud paradigm to meet with different challenges posed by Global Software Development (GSD). We are suggesting that this will result in GSD benefitting from the cloud's infrastructure, platform, and provision of software as a service features. Information and data on the cloud is transmitted and shared by means of web services which work on underlying Service Oriented Architecture (SOA) principle.

We argue that the cloud paradigm has the potential to turn over a few more unturned stones of GSD issues which are a significant hurdle for the development of successful projects in the GSD situation. But, we are planning to develop our ideas further with a view to filling the gap between technical proficiency and meeting the needs of developers. So, like SOA, we cannot expect the cloud paradigm to address some psychological and social issues like *trust* but we can reduce their negative impact through the use of this model.

ACKNOWLEDGMENT T

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube). It was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero – the Irish Software Engineering Research Centre (www.lero.ie), and a Higher Education Authority grant PRTLI 4 to the Lero Graduate School in Software Engineering.

REFERENCES

[1] J. D. Herbsleb, "Global software engineering: the future of socio- technical coordination," in Proceedings of the Future of Software Engineering (FOSE'07), 2007, pp. 188-198.

[2] R.E. Grinter, J.D. Herbsleb, and D.E. Perry, "The geography of coordination: dealing with distance in R&D work," in Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '99), ACM Press, New York, 1999, pp. 306–315.

[3] J. A. O'Brien, "Management information systems: managing information technology in the business enterprise," Mc Graw Hill Irwin, 2002.

[4] S. S. Toaff, "Don't play with "Mouths of Fire," and other lessons of global software development." Cutter IT Journal 15(11), 2002, pp. 23- 28.

[5] E. Carmel and P. Tjia, "Offshoring information technology: sourcing and outsourcing to a global workforce." Cambridge, UK, Cambridge University Press, 2005.

[6] A. J. Espinosa and E. Carmel, "The impact of time separation on coordination in global software teams: a conceptual foundation" Software Process Improvement and Practice, 8(4), 2003, pp. 249 – 266.

[7] A. Sarma and V. D. Hoek, "Towards awareness in the large," in Proceedings of the International Conference on Global Software Engineering, 2006, pp. 127-31.

[8] E. Carmel, “global software teams: collaboration across borders and time zones”. Saddle River, NJ, Prentice Hall, 1999.

[9] D. W. Karolak, “Global software development: managing virtual teams and environments.” Los Alamitos, CA, USA, IEEE Computer Society Press, 1999.

[10] J. D. Herbsleb and D. Moitra, "Global software development." IEEE Software 18(2), 2001, pp. 16-20.

[11] V. Clerc, P. Lago, and H. V. Vliet, “The architect’s mindset,” 3rd International Conference on the Quality of Software Architectures, Volume 4880 of Lecture Notes in Computer Science, 2007, pp. 231-249, Springer Berlin / Heidelberg.

[12] E. Carmel, “Building your information systems from the other side of the world: how Infosys manages time differences,” Management Information Systems Quarterly -MIS Quarterly Executive, vol. 5, no. 1, Mar 2006, pp. 43-53.

[13] V. Casey, S. Deshpande, and I. Richardson, “Outsourcing and offshoring software development: the remote developers’ perspective,” Global Sourcing Workshop, Val d’Isere, France March 2008.

[14] F. Lanubile, D. Damian, and H. L. Oppenheimer, "Global software development: technical, organizational, and social challenges," SIGSOFT Software Engineering Notes 28(6): 1 - 4.

[15] R. T. Watson, T. H. Ho, and K. S. Raman, “Culture: a fourth dimension of group support systems,” Communications of the ACM, 37-10, 1994, pp. 44-55.

[16] C. M. Beise, “IT project management and virtual teams,” in Proceedings of the 2004 SIGMIS Conference on Computer Personnel Research: Careers, Culture, and Ethics in A Networked Environment (Tucson, AZ, USA, April 22 - 24, 2004). SIGMIS CPR '04. ACM, New York, NY, pp. 129-133.

[17] J. Yang and Z. Chen, "Cloud computing research and security issues," International Conference on Computational Intelligence and Software Engineering *(*CiSE*),* 2010, pp. 1-3.

[18] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service," Computer, vol.36, no.10, Oct. 2003, pp. 38- 44.

[19] S.-o Setamanit, W. Wakeland, and D. Raffo, "Improving global software development project performance using simulation," Portland International Center for Management of Engineering and Technology, 5-9 August, 2007, pp. 2458-2466.

[20] K. Zhang, X. Zhang, W. Sun, H. Liang, Y. Huang, L. Zeng, and X. Liu, "A policy-driven approach for software-as-services customization," in Proceedings of the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, CEC/EEE 2007, pp.123-130.

[21] A. Bertolino and A. Polini, “SOA test governance: enabling service integration across organization and technology borders”, IEEE International Conference on Software Testing V erification and Validation Workshops.

[22] L. Osterweil, “Software processes are software too,” in Proceedings of the 9th International Conference on Software Engineering, IEEE Computer Society Press Los Alamtos, CA, USA.

[23] S. Dustdar and W. Schreiner, “A survey on web services composition”, International Journal of Web and Grid Services, vol. 1, no. 1, 2005, pp. 1-30.

[24] L. Cheng, C. DeSouza, S. Hupfer, J. Patterson, and S. Ross, “Building cllaboration into IDEs,” ACM Queue, vol.1, no.9, 2004, pp. 40-50.

[25] A. Sarma, Z. Noroozi, and A. V. D. Hoek, “Palantír: raising awareness among configuration management workspaces,” in Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 444-454.

[26] S. Zhang; S. Zhang, X. Chen, and X. Huo, "Cloud computing research and development trend," in Proceedings of the 2nd International Conference on Future Networks, 2010, pp. 93-97.

[27] J. Yang and Z. Chen, "Cloud computing research and security issues," in Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE), 2010, pp. 1-3.

[28] T.J. Allen, "Managing the flow of technology," Cambridge, MA: MIT Press, 1977.

[29] R. Martignoni, "Global sourcing of software development – a review of tools and services", in Proceedings of the 4th IEEE International Conference on Global Software Engineering, 2009, pp. 303-308.

[30] M. P. Papazoglou, "Web Services and Business Transactions", World Wide Web, vol. 6, no. 1, March 2003, pp. 49-91.

[31] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, and S. Thatte, "Web services transaction specifications," available online, last accessed 16 August, 2005: http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/

[32] F. Cabrera, et al., "Web services coordination (WS-coordination)," Version 1.0, available online, last accessed August 2005: ftp://www6.software.ibm.com/software/developer/library/ws-coordination.pdf

[33] D. Bunting, M.C.O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson, "Web services transaction management" (WS- TXM), V er1.0. http://developers.sun.com/techtopics/webservices/wscaf/wstxm.pdf

[34] D. Cubranic and G. Murphy, "Hipikat: recommending pertinent software development artifacts," in Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 408-418.

[35] M. Jensen, S. Menon, L.E. Mangset, and V. Dalberg, "Managing offshore outsourcing of knowledge intensive projects – A people centric approach," in Proceedings of the 2nd International Conference on Global Software Engineering, 2007, pp. 186-196.

[36] C.R.B.D.Souza,D.Redmiles,L.T.Cheng,D.Millen,andJ.Patterson, "Sometimes you need to see through walls: a field study of application programming interfaces," in Proceedings of the 2004 ACM conference on Computer supported cooperative work, pp. 63-71.

[37] J. D. Herbsleb and R. E. Grinter, "Splitting the organization and integrating the code: Conway's law revisted," in Proceedings of the 21st International Conference on Software Engineering, 1999, pp. 85-95.

[38] L. W. Pires, L. F. Wombacher, A. V. Sinderen, and M. J. Chihung Chi, "Stakeholder interactions to support service creation in cloud computing," in Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010, pp. 173-176.

[39] Armbrust et. al., "Above the clouds: a Berkley view of cloud computing," EECS Department, University of California, Berkeley Technical Report No. UCB/EECS-2009-28 February 10, 2009.

[40] E. Kit, "Software testing in the real world: improving the process", Addison-Wesley, Reading, MA, USA, 1995.

[41] H. Do and G. Rothermel, "An empirical study of regression testing techniques incorporating context and lifetime factors and improved cost- benefit models," in Proceedings of 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2006, pp. 141-151.

[42] J. Kasurinen, "Elaborating software test processes and strategies," in Proceedings of the 3rd International Conference on Software Testing, V erification, and V alidation, 2010, pp. 355-358.

[43] L. J. Zhang, "EIC editorial: introduction to the body of knowledge areas of services computing," IEEE Transactions on Service Computing, vol. 1, no. 2, April-June, 2008, pp. 62-74.

[44] L. J. Zhang, J. Zhang, and H. Cai, "Services computing, core enabling technology of the modern services industry," published by Springer and Tsinghua University Press, 2007.

[45] L. J. Zhang and Q. Zhou, "CCOA: cloud computing open architecture," in Proceedings of the 7th IEEE International Conference on Web Services, 2009, pp. 607-616.

[46] R. Guha, and D. Al-Dabass, "Impact of Web 2.0 and cloud computing platform on Software Engineering," in Proceedings of the International Symposium on Electronic System Design (ISED), 2010, pp.213-218.

[47] J. Schaper, "Cloud Services, " in Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies, 2010, pp. 91.

[48] A. Elfatatry and P. Layzell, "Software as a service: a negotiation perspective," in proceedings of the 26th International Conference on Computer Software and Applications Conference, COMPSAC 2002. pp. 501-506.

[49] A. Avritzer, D. Paulish, Y. Cai, and K. Sethi, "Coordination implications of software architecture in a global software development project," Journal of Systems and Software 83(10), pp. 1881-1895.

[50] P. Hartman, "ESB enablement of an international corporate acquisition, an experience report," in Proceedings of the 3rd IEEE International Conference on Global Software Engineering, 2008, pp. 200-204.

[51] M. D. Dikaiakos, G. Pallis, D. Katsaros, P. Mehra, and A. Vakali, "Cloud computing: distributed internet computing for IT and scientific research," Internet Computing, vol. 13, no. 5, pp. 10-13.

[52] E. E. Jennings, "Routes to the executive sites," New York: Mcgraw-Hill, 1971.

[53] S. L. Jarvenpaa and D. E. Leidner, "Communication and trust in global teams," Journal of Computer Mediated Communication, vol. 10, no. 6, 1999, pp. 791-815.

[54] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and privacy in cloud computing: a survey," in Proceedings of the 6th International Conference on Semantics, Knowledge, and Grids, 2010, pp. 105-112.

[55] L. Badger, T. Grance, R. P.-Comer, J. Voas, "Draft cloud computing synopsis and recommendations," National Institute of Standards and Technology, Special Publication 800-146, May 2011.

[56] http://itsm.certification.info/ownership.html

[57] A.Begel,N.Nagappan,C.Poile,L.andLayman,"Coordinationinlarge- scale software teams," ICSE Workshop on Cooperative and Human Aspects on Software Engineering, 2009, CHASE '09, pp. 1-7.

[58] A.Begel,N.Nagappan,"Globalsoftwaredevelopment:whodoesit?,"in Proceedings of the 3rd IEEE International Conference on Global Software Engineering, 2008. pp.195-199.

[59] M. Behrendt, et al., "Introduction and architecture overview: IBM cloud computing reference architecture 2.0," Draft Version V1.0, 2011.

[60] T. Nguyen, T. Wolf, and D. Damian, "Global software development and delay: does distance still matter?," in Proceedings of the 3rd IEEE International Conference on Global Software Engineering, 2008, pp. 45- 54.

[61] P. Mohapatra, P. Bjorndal, and K. Smiley, "Causal analysis of factors governing collaboration in global software development teams," in Proceedings of the 5th IEEE International Conference on Global Software Engineering, 2010, pp. 128 – 132.

[62] T. Niinimaki, A. Piri, and C. Lassenius, "Factors affecting audio and text-based communication media choice in global software development projects", in Proceedings of the 4th IEEE International Conference on Global Software Engineering, 2009, pp. 153-162.

[63] M. Cataldo, M. Bass, J. D. Herbsleb, and L. Bass, "Factors affecting audio and text-based communication media choice in global software development projects", in Proceedings of the 4th IEEE International Conference on Global Software Engineering, 2009, pp. 153-162.

[64] V.Casey,“Leveragingorexploitingculturaldifference?”,inProceedings of the 4th IEEE International Conference on Global Software Engineering, 2009, pp. 8 - 17 .

[65] V. Casey and I. Richardson, “Project management within virtual software teams”, in Proceedings of the International Conference on Global Software Engineering, 2006, pp. 33-42.

[66] D. Damian, F. Lanubile, and H. L. Oppenheimer, “Addressing the challenges of software industry globalization- the workshop on global software development,” in Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 793-794.

[67] R. Heeks, S. Krishna, B. Nicholsen, and S. Sahay,” Synching or sinking- global software outsourcing relationships,” IEEE Software, vol. 18, Issue. 2 , 2001, pp. 54 - 60.

[68] H. Klein, A. Rausch, and E. Fischer, “Process-based collaboration in global software engineering,” in Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009, pp. 263 - 266.

[69] H. K. Edwards and V. Sridhar, “Analysis of the efectiveness of global virtual teams in software engineering projects,” in Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.

[70] J. Portillo-Rodriguez, A. Vizcaino, C. Ebert, and M. Piattini “Tools to support global software development processes: a survey”, in Proceedings of the 5th IEEE International Conference on Global Software Engineering, 2010, pp. 13 - 22.

1

Appendix F: Going Global with Agile Service Networks

Damian A. Tamburri advised by Patricia Lago and Hans Van Vliet

Abstract—ASNs are emergent networks of service-based applications (nodes) which collaborate through agile (i.e. adaptable) transactions. GSE comprises the management of project teams distanced in both space and time, collaborating in the same development effort. The GSE condition poses challenges both technical (e.g. geolocalization of resources, information continuity between timezones, etc.) and social (e.g. collaboration between different cultures, fear of competition, etc.). ASNs can be used to build an adaptable social network (ASNGSE) supporting the collaborations (edges of ASNGSE) of GSE teams (nodes of ASNGSE).Agile Service Networks can be used to support Global Software Engineering (GSE).

I. INTRODUCTION

GSE is a software engineering strategy in which organizations try to achieve round-the-clock productivity by distributing production along different continents and timezones [10], [11], [19]. GSE is a business decision. As a consequence of this decision, teams' time and space distance pose challenges both social and technical. Social challenges, such as cultural barriers, language difference, social class, unawareness of others limit the degree to which teams involved in GSE are able to share information effectively and collaborate efficiently. Technical challenges, such as space distance, information discontinuity between timezones, geolocalization of development resources hinder the way productivity can be coordinated on a global scale. These challenges inhibit teams' communication and collaboration to a point where project failure risk increases [11], [8], [19]. Major causes for delays and cost explosion in GSE projects are miscommunication on project requirements, deadlines and architecture [4]. To exemplify these issues consider the following simple (and very common) scenario, from a real- life situation [5], [9]:

"An unavoidable technical constraint discovered by a programmer causes design to be reworked, designers negotiate an adjustment in requirements with system analysts and modify their designs appropriately. Subsequently, this technical constraint is raised repeatedly by other programmers who were not informed of the design changes (or are using designs that may have been unrelated to the adjusted design, but related to the affected requirement). This chaotic disruption wastes time and effort and causes unnecessary aggravation to the development team."

GSE technical difficulties should be supported through socially-enabled technologies for the above challenges to be overcome [11]. These technologies would be able to avert chaos by automatically linking together the people involved, increasing their awareness. ASNs are emerging networks of collaborative service applications (nodes) which interoperate through agile transactions (edges), i.e. transactions which react to context change dynamically adapting themselves [2]. ASNs stem from the business decision of corporate collaborations to be formed globally, in pursuit of business gain for the nodes involved (i.e. every corporate partner) [6]. A fundamental similarity can be identified between GSE and ASNs. Both stem from business decisions. Moreover, a crucial complementarity exists between them. On one side, GSE needs dynamism among nodes (development teams) and their collaboration towards business gain (timely delivery). On the other, ASNs are supporting dynamic collaborations among nodes which are teaming up to increase business gain. These relations led us to consider three potentials of ASNs: (a) Collaborativeness as well as context adaptability make ASNs ideal to develop flexible social- networking tools for GSE; (b) being service-based, ASNs can be used in the cloud, using its services to further support GSE; (c) ASNs' emergence makes them ideal to build dynamic communication tools against distance in time and space. Based on these considerations, we argue that GSE challenges can be overcome through an ASN-based social network (ASNGSE) providing agility of communications and collaborations (edges of ASNGSE) to globally located IT professionals (nodes of ASNGSE). Global professionals can be represented as nodes in an agile (i.e.

adaptable and emergent) organizational social network to deliver the final product, just-in-time and sufficiently-good. The next section provides detail of the research design and challenges for this proposal. Section IV provides the main contributions we plan to obtain; section II recaps related work. Finally, section V concludes this paper1.

II. RELATED WORK

We find our efforts most similar to [12], [16], [11] and [7], both in terms of intents and motivation.

In [12] authors represent the industrial/business perspective on digital formations and global development; [16] and [11] both provide empirical studies in GSE that stress the presence of social factors which shape the outcome of global development projects. Finally, [7], [15] provides the previous exploration of social networking technologies as a service network.

In [12] authors examine political, social, cultural and economic implication of the digital revolution. They explore their influences in modern organizational activities such as business planning, project management, etc. They conclude that digital formations (i.e. new social structures which emerge in digitally-supported societies) should be considered a frontier for further research and development efforts. From the premises and conclusions of this work we draw motivation for our effort, since we try to research into organizational social structures (a concept from sociology) and use these results to shape a socially-enabled technology.

In [16] authors recall delay as a typical problem to be tackled in GSE. They explore this problem, its implication in GSE as well as its actuality. Empirical data from IBM leads the authors to observe that indeed distance and (mishandled) collaboration are crisis causes in GSE. The authors use social-network analysis to obtain data on collaboration. Our (empirical) plan is similar to such research. Our approach is top-down since we try to build a social-network and validate its functions in real-life GSE scenarios, rather than bottom- up by exploring GSE industrial scenarios to reverse engineer requirements. The same observations can be made for [11], since the authors operate in a manner similar to [16]. More in particular in [11], authors provide us motivations to invest in supporting the GSE OSS since the authors conclude that social networks can play a serious role in increasing the mutual awareness of professionals involved in GSE.

Finally, in [7], quoting the authors, "[the aim is] a frame- work integrating Web 2.0 social aspects to automatically enrich Web services with semantic meanings based on community consensus". In much the same way. we propose to use both technical and social technologies to enrich the current collaboration possibilities among GSE professionals, through an agile service social network. In earlier works such as [15], the same idea was posted as a potential solution to the on-set of the global collaboration needs in our current market.

III. RESEARCH DESIGN

This section provides two elements: (a) the problem we want to tackle; (b) the description of the solution we advocate.

A. Problem Statement

Distances in time and space make it impossible for GSE teams to communicate and coordinate their effort in an efficient manner [11]. Technical challenges, such as information discontinuity between timezones, limit the advantages of GSE. Social challenges, such as language difference and social class difference, inhibit collaborativeness among GSE teams, increasing its chances of failure [19]. How can ASNs be used to support GSE?

B. Our Proposed Solution

The solution we advocate is an ASN-based social net- work (ASNGSE). This technology should exhibit four key characteristics:

1) Agile context awareness. ASNs are able to detect changes in the context and dynamically support different scenarios as needed. In GSE for instance, round-the-clock productivity could be supported by dynamically allocating collaborations between teams, by modeling each developer as a set of skills and allowing for their seamless (re-)allocation based on their timezone, location and needs. Also,

seamless hand- off of relevant informations between two contiguous timezones could be used to ease the coordination of sequential or dependent work packages.

2) Deployed in the cloud. Cloud computing, has potentials which fit with GSE needs [14], [20]. For instance, GSE resources rendered available in the cloud allow for rapid resource location and access on a global scale. Also, communication and information continuity between timezones may be requested as needed.

3) Satisfying GSE social requirements. GSE teams together create an Organizational Social Structure (OSS) [17], part of a global corporation. Social interactions in OSSs depend on personal- or corporate-specific practices, which include work habits, methods, technologies to support cooperation, etc. In GSE, for instance, supporting social interactions among developers from different companies and cultures, would require letting them use own tools, languages and own methods seamlessly. ASNs can help in doing that through adaptable creation of service compositions or transparent information proxying (i.e. providing seamless switching between answering nodes in case of difficulty). Another example could be using ASNs to autonomously try and assemble a common tool- workbench for all teams. Lastly, an ASN to support GSD could compute the work allocation to teams, using (up-to-date and context-aware information) on project requirements, time constraints, current availability, etc.

4) Project-centric as well as People-centric. Enterprise Social Networking technologies already exist which could potentially represent (and supporting) the social network of an entire corporation. What is still missing is the dynamic / automatic adjustment of its granular- ity, to support the global software development project (against its changing context) as well as the people involved (e.g. technicians, developers, managers, etc.). Moreover, none of these technologies provide flexible and adaptable collaboration channels (e.g. adaptable status-tracking, always-on reachability of key roles, worldwide project chatter, etc.) among professionals collaborating in the same GSE effort.

Figure 1 tries to visualize the concept we have in mind. In our vision different globally-distributed teams (upper part) can be linked with devices (e.g. computers, in the lower part). The network links professionals (below) but brings them together into an emergent social network (above) which supports their collaboration as well as their sense of participation to a project. This can support them in coordinating their effort or prevent disaster scenarios.

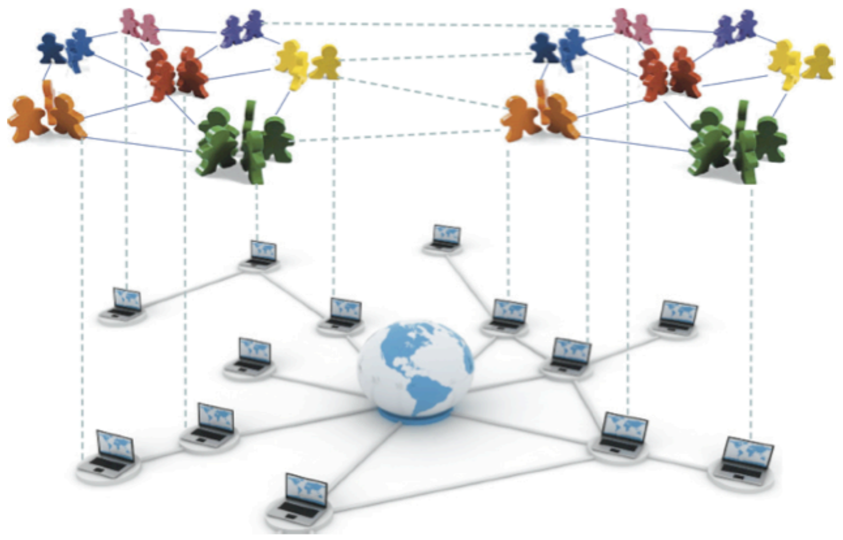


Figure 1. ASNs for GSD

With respect to the example in section I, the solution we have in mind would make the global software engineering network, more "agile" in three dimensions: (a) awareness - every node would be proactively aware of the status and presence of all others; (b) emergence - ASNs could increase-decrease their functionality through cloud, adding and removing capabilities, devices, people and commodities as needed; (c) organization - our ASNs would support the OSS of the current GSE, adapting it if needed, thereby strengthening the organizational efficiency of GSE efforts, by virtue of Conway's law [3].

C. Research Methods

To develop a solution with the mentioned characteristics, we plan the following actions: 1) A systematic Literature Review (SLR) into OSSs. This review provides the requirements, features and potentials for social structures present in literature. Mapping GSE on these results can give us the profile of the OSS for GSE, as well as a picture of its context.

2) Case-study research on ASNs. We have to understand the potentials and limits of the context-awareness and adaptation mechanisms behind ASNs. For this we plan to study cases from domains in which dynamic adaptation is of extreme importance (e.g. emergency management). These studies can give us a feasibility assessment of different adaptation mechanisms avail- able for ASNs.

3) Context-Modeling. To allow ASNs' adaptability to the GSE context, this must be clearly developed and formalized. Case-studies, interviews or other empirical research in industries involved in GSE, can be used to validate the obtained context-model. From a validated context-model a prototype of the social network can be developed and proposed in industry to be validated through action research.

4) Feasibility studies. To understand ASNs' feasibility in the cloud, a requirements-based feasibility study of ASNs in the cloud must be carried out.

5) Prototyping. A prototype of ASN to support the GSE OSS must be developed to evaluate its practical value against our objectives. The prototype should model and support (at least a portion of) the GSE OSS, as well as adapting to (at least a portion of) its context's changes.

6) Validation through action research. We are already in contact with large industrial partners to validate the a working prototype, using standard 3-to-6-month cycles of action research, across a three-year period. A working prototype will be deployed and evaluated in real industrial scenarios.

So far (8 months of investigation) we have obtained two key results: (a) we have established the feasibility of ASNs in the cloud [21], [18]; (b) we have sketched the very first version of a context-model for GSE [23]. Currently, we are finishing a systematic survey into OSSs [17].

IV. MAIN CONTRIBUTIONS

The dissertation about this research is expected to offer the following key contributions: (a) a clear-cut definition of ASNs, stating its defining characteristics, operational me- chanics as well as (certified) applicative grounds (e.g. cloud computing and emergency management through case study research, GSE, etc.); (b) a validated OSS profile and context- model for GSE including social structure requirements, so that ad-hoc support tools may be (further) developed for it; (c) an ASN-based social-network prototype to support collaboration in GSE. The prototype would be validated in practice through action research.

V. CONCLUSION

In this paper we outlined the proposal of an ASN-based social network (ASNGSE). We elaborated this proposal into a research plan. We have pointed towards preliminary results which are encouraging us in exploring the research path futher. Specifically, by analyzing GSE and matching its challenges with ASNs we concluded that indeed ASNs are supportive [21]. Moreover, by inspecting GSE literature we were able to draw an initial version of a GSE context-model [23]. In the future we plan to increase this context-model with social requirements from an SLR we are conducting. Finally, we plan to develop a very initial version of a prototype for the proposed social-networking ASN for GSE, to initiate industrial action-research validation.

LIST OF MY PREVIOUS PUBLICATIONS

I was an active author and main contributor in [13], [22], [1], [18], [23], [21]. The last three works are resulting from the exploration of the idea presented in this symposium paper.

REFERENCES

[1] Marco Autili, Paolo Di Benedetto, Paola Inverardi, and Damien A. Tamburri. Towards self-evolving context-aware services. ECEASST, 11, 2008.

[2] Noel Carroll, Eoin Whelan, and Ita Richardson. Applying social network analysis to discover service innovation within agile service networks. Service Science, 2:pp 225–244, November 2010.

[3] M.E. Conway. How do committees invent. Datamation, 14(4):28–31, 1968.

[4] Daniela Damian. Stakeholders in global requirements engi- neering: Lessons learned from practice. IEEE Software.

[5] HeatherOppenheimerDanielaDamian,FilippoLanubile.The international workshop on global software development. In Proceedings of ICSE Conference 2003, 2003.

[6] Shoumen Datta, Bob Betts, Mark Dinning, Feryal Erhun, Tom Gibbs, Pinar Keskinocak, Hui Li, Mike Li, and Micah Samuels. Adaptive value networks. In Yoon S. Chang, Harris C. Makatsoris, and Howard D. Richards, editors, Evolution of Supply Chain Management, pages 3–67. Springer US, 2004.

[7] Khaled El-Goarany, Iman Saleh, and Gregory Kulczycki. The social service network - web 2.0 can make semantic web services happen. In Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pages 419–423, Washington, DC, USA, 2008. IEEE Computer Society.

[8] The Standish Group. Standish Chaos Report. 1995-2009.

[9] Elizabeth Hargreaves, Daniela Damian, Filippo Lanubile, and James Chisan. Global software development: building a research community. SIGSOFT Softw. Eng. Notes, 29:1–5, September 2004.

[10] James D. Herbsleb. Global software engineering: The future of socio-technical coordination. In Lionel C. Briand and Alexander L. Wolf, editors, FOSE, pages 188–198, 2007.

[11] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: distance and speed. In ICSE '01: Proceedings of the 23rd International Conference on Software Engineering, pages 81–90, Washington, DC, USA, 2001. IEEE Computer Society.

[12] Robert Latham and Saskia Sassen, editors. Digital forma- tions: IT and new architectures in the global realm. Princeton University Press, Princeton, NJ, 2005.

[13] Ivano Malavolta, Henry Muccini, Patrizio Pelliccione, and Damian A. Tamburri. Providing architectural languages and tools interoperability through model transformation technolo- gies. IEEE Transactions on Software Engineering, 36(1):119– 140, January 2010.

[14] Ivanka Menken and Gerard Blokdijk. Cloud Computing Best Practice Guide: Strategies, Methods and Challenges To Managing Services in the Cloud. Emereo Pty Ltd, London, UK, UK, 2009.

[15] Alexander Mikroyannidis. Toward a social semantic web. IEEE Computer, 40(11):113–115, 2007.

[16] Thanh Nguyen, Timo Wolf, and Daniela Damian. Global software development and delay: Does distance still matter? Global Software Engineering, 2008. ICGSE 2008. IEEE In- ternational Conference on, pages 45–54, Aug. 2008.

[17] K. Ruikar, L. Koskela, and M. Sexton. Communities of practice in construction case study organisations: Questions and insights. Construction Innovation, 9(4):434–, 2009.

[18] Maryam Razavian Christina Manteli Patricia Lago Elisabetta Di Nitto Sajid Hashmi, Viktor Clerc, Ita Richardson, and Damian A. Tamburri. Software as a service in the cloud to support global software development. In 5th International Workshop on Tool Support and Requirements Management in Distributed Projects (REMIDI'11). 6th International Confer- ence on Global Software Engineering (ICGSE 2011).

[19] Raghvinder Sangwan, Matthew Bass, Neel Mullick, Daniel J. Paulish, and Juergen Kazmeier. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications, Boston, MA, USA, 2006.

[20] Stefan Tai, Jens Nimis, Alexander Lenk, and Markus Klems. Cloud service engineering. In Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastin Uchitel, editors, ICSE (2), pages 475–476. ACM, 2010.

[21] Damian A. Tamburri and Patricia Lago. Satisfying cloud computing requirements through agile service networks. Pro- ceedings of SERVICES 2011, 2011.

[22] Damian A. Tamburri, Patricia Lago, and Henry Muccini. Leveraging software architectures through the iso/iec 42010 standard: A feasibility study. In TEAR, pages 71–85, 2010.

[23] Lago P. Tamburri, D. A. Supporting communication and cooperation in global software development with agile service networks. ECSA 2011, Springer:8 pages, 2011.

Appendix G: On the Nature of GSE Organizational Social Structures: an Empirical Study

Damian A. Tamburri∗, Elisabetta Di Nitto†, Patricia Lago∗, and Hans Van Vliet∗

***Abstract*—In Global Software Engineering (GSE), people are organized in teams, distanced in space, time and culture. Organizational research calls this interplay of people an Orga- nizational Social Structure (OSS). Previous literature in GSE shows that its OSS is highly dynamic and unpredictable. This paper presents a mapping of OSS types on GSE organizational factors, based on empirical evidence. We made two observa- tions: first, current OSS types dont support factors related to GSE process management and organizational efficiency (e.g. risk management, language, etc.). Second, OSSs in GSE have attributes which dont map to any GSE factor, rather introduce a new one, awareness management (e.g. awareness of skills to others, awareness of tasks, tasks (re-)localization, etc.). Our conclusions are twofold. On one hand, OSSs for GSE should focus on increasing support to process management and organizational efficiency. On the other hand, research in GSE should include factors focusing on awareness management.**

***Keywords*-Global Software Development, Social Computing, Social Structures, Requirements Engineering, Human Factors, Empirical Study**

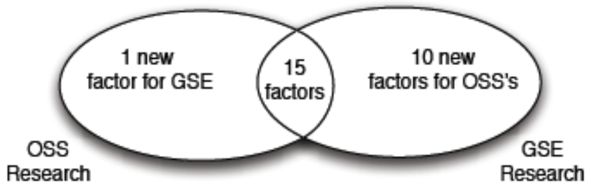
Global Software Engineering (GSE) is a business decision entailing project teams to collaborate globally on the same project, from different timezones [7], [8]. Literature shows how this decision increases failure risks [9], [6]. Different factors cause increased risks. For example, distance in space and time makes (cross-)dependencies among project tasks very tight; in this web of dependencies a single unmet deadline can cause a ripple effect compromising the whole project [16]. Also, the cultural distance among developers in different cultural areas (e.g. Europe vs. India) can cause fear, mistrust or other “social” problems which hinder communication and collaboration [4].

An Organizational Social Structure (OSS) represents the emergent web of (social) ties, practices and cognitive approaches between individuals collaborating towards a common goal [19]. An OSS’ purpose is to enable the creation and sharing of knowledge between individuals so that the final goal can be reached [2]. Within GSE distributed teams collaborate (i.e. they share knowledge) to develop software systems (i.e. their final goal): by definition, this constitutes an OSS. On the one hand, GSE incurs an OSS that fits the type of people interactions that naturally arise in that context. On the other hand, results in OSS can help us understand, and improve, the challenges of GSE [8].

In this paper we map the current state of the art in OSS’s on current practice in GSE. The current state of the art in OSS’s is derived from a systematic literature review [17], while the current practice in GSE is derived from empirical research reported in [4], [15].

The empirical research in [4], [15] identifies 25 organizational factors that an organization has to decide on when embarking on a GSE project. 10 of these factors are not matched by any OSS discussed in literature. These 10 factors relate to process management and organizational efficiency. Conversely, the OSS’s that best fit GSE have a number of attributes that address awareness management (e.g. awareness of people, awareness of their skills, of their allotted tasks, awareness of tasks (re-)localization as needed, etc.). This factor, namely, awareness management, is not mentioned in [4], [15].

Figure 1. Our Results: OSSs miss 10 GSE factors, and introduce 1 new factor.



Our conclusions are twofold. On one hand, research in OSS's for GSE should focus on increasing its support to "process management" and "organizational efficiency" factors. On the other hand, research in GSE should include factors focusing on awareness management.

The rest of the paper is structured as follows: section II-B provides an overview of the materials we used for this study (OSSs and GSE organizational Factors) as well as the results we obtained in mapping them. Section III provides discussions and observations on results. Finally, section IV concludes the paper hinting to future work.

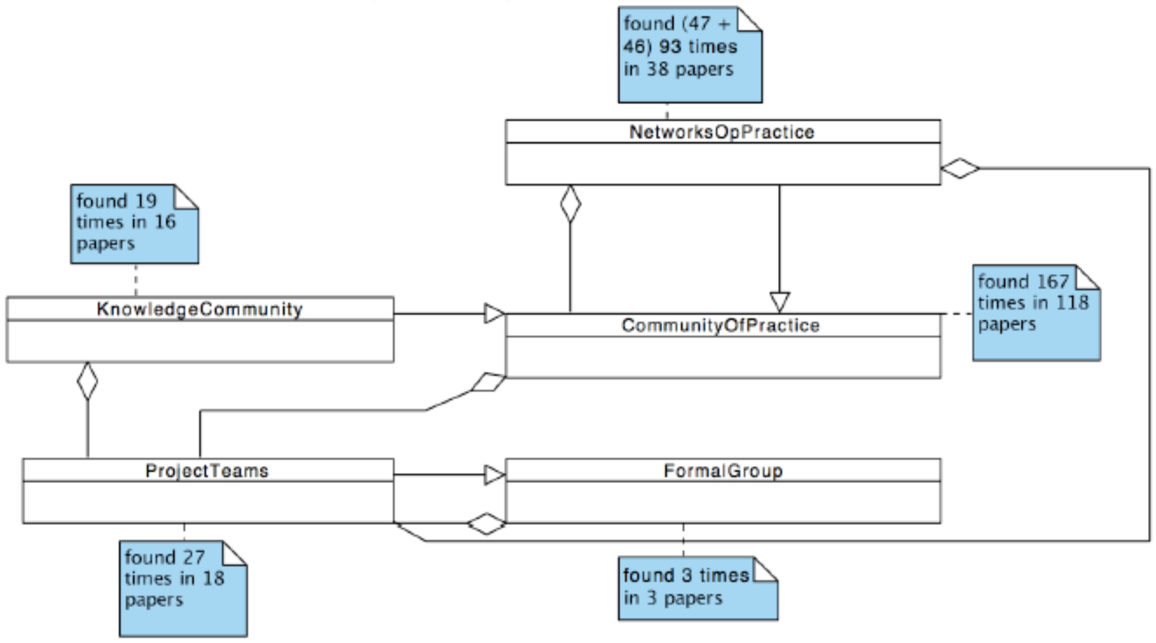
II. RESEARCH APPROACH

A. What we used

Two key contributions were used for the work in this paper. The first is a systematic literature review (SLR) we conducted into OSSs. The second is published in [4] and [15].

We conducted an SLR [17] to obtain the state of the art in OSS's. Using a grounded-theory approach to study the literature [12] we obtained 13 types of OSSs together with their defining attributes. We narrowed down to the types most relevant to the GSE domain by comparing the 13 OSS types with definitions of GSE from [8], [7], [16] and [6]. We obtained 5 types: Project Teams, Networks of Practice, Knowledge Communities, Communities of Practice and Formal Groups. Figure 2 captures their mutual relations.

Figure 2. OSS types relevant in GSE domains


found (47 + 46) 93 times in 38 papers
NetworksOpPractice
found 19 times in 16 papers
KnowledgeCommunity
CommunityOfPractice
found 167 times in 118 papers
ProjectTeams
FormalGroup
found 27 times in 18 papers
found 3 times in 3 papers

bound together by shared expertise and passion for a joint enterprise – engineers engaged in deep-water drilling, for example, consultants who specialize in strategic marketing, or frontline managers in charge of check processing at a large commercial bank". A CoP consists of co-located groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting frequently and in the same geolocation. Therefore CoPs serve as scaffolding for organizational learning in one specific practice. For a CoP to take place, a vital requirement must be satisfied: co-location. All developers have to meet in the same place, at the same time for the dynamics of CoPs to take place. For example, Software Architects' workshops in GSE projects can be seen as an instance of CoPs in GSE, since they entail, co-located professionals to share a common practice (Software Architecture) for the benefit of the project. Instances of this type in practice are (co-located) software architects' meetings: these are common in GSE to synchronize efforts and plan further progress.

2) Networks of Practice (NoP): Quoting from [11] "NoP comprises a larger, geographically dispersed group of participants engaged in a shared practice or common topic of interest [...] CoPs and NoPs share the characteristics of being emergent and self-organizing, and the participants create communication linkages inside and between organizations that provide an "invisible" net existing beside the formal organizational hierarchy". A NoP is a networked system of communication and collaboration that connects CoPs (which are localized). In principle anyone can join it without selection of candidates (e.g. an OpenSource forge, like SourceForge, is an instance of NoP). NoPs have a high geodispersion, i.e. they can span geographical and time distances alike. This increases their visibility and the reachability by members. An unspoken requirement for entry is the expected IT literacy of members. IT literacy must be high since the tools needed to take part in NoPs are IT-based (e.g. Micro-blogs, forums, hang-outs, etc.). NoPs are composed of CoPs (which are co-located). They inherit from CoPs the enforcement of shared repositories of knowledge for their members, as well as the presence of a common practice acting as an engagement within the network. Differently than CoPs, NoPs can be seen as IT-enabled global networks, since their chief aim is to allow communication (and collaboration) on the same practice through large geographical distance. For example, each GSE team (e.g. people, skills, documents, etc.) can be seen as a node within a GSE NoP [18].

3) Formal Groups (FG): FGs, are exemplified in [10] as "[groups of] teams and/or workgroups [. . .]. Numerous different definitions of diversity have been put forth; however, they generally distinguish between two main sets of characteristics [for FGs]: 1) diversity of observable or visible detectable attributes such as ethnic background, age, and gender; 2) diversity with respect to non-observable, less visible or underlying attributes such as knowledge disciplines and business experiences". FGs a set of people which is explicitly grouped by corporations to act on (or by means of) them (e.g. governing employees or ease their job or practice, by grouping them in areas of interests). Each group has a single organizational goal (governing boards are groups of executives whose goal is to devise and apply governance practices). In comparison to other types, they seldom rely on networking technologies to link their members, on the contrary, they are local in nature. Moreover, it is very common for organizations to have these groups and extract project teams out of them (and therefore they are composed of project teams). Moreover, since project teams are instances of formal groups but tailored specifically to solve a particular problem, they inherit organizational aspects of formal groups such as clear-cut definition of tasks, complementary set of skills, etc. A perfect example of an FG is the JPL (Jet Propulsion Lab) within NASA. An example in the GSE domain can be seen in the SCR group at Siemens, in which Siemens researchers, work collaboratively to develop best practices [14]. Formal groups are very similar to the organizational units, or "sites", which are used in GSE.

4) Knowledge Communities (KC): Quoting from [5] "Virtual knowledge communities are organized groups of experts and other interested parties, who exchange knowledge on their field of expertise or knowledge domain in cyberspace within and across corporate and geographical borders. Virtual knowledge communities focus on their knowledge domain and over time expand their expertise through collaboration. They interact around relevant issues and build a common knowledge base". Essentially KCs, are groups of people with a shared passion to create, use, and share new knowledge for tangible business purposes (e.g. increased sales, increased product offer, clients profiling, etc.). The main difference with other types is in their specific tie to precise business goals for the organizational sponsor. Moreover, they are not limited to use electronic communication and collaboration means (such as NoPs) but rather they inherit from CoPs the enforcement of co-located meetings or workshops to devise or explore new ideas. Specific industrial groups such as the JDA global alliance program 1, focused on supply-chains, can be seen as a knowledge community for GSE, since they focus on best practices and knowledge interchange around problems specific to (a specific domain of) GSE.

5) Project Teams (PT): Lindkvist [13] provides a general definition of PTs with the following words: "[PTs are] temporary organizations or project groups within firms [that] consist of people, most of whom have not met before, who have to engage in swift socialization and carry out a pre-specified

task within set limits as to time and costs. Moreover, they comprise a mix of individuals with highly specialized competences, making it difficult to establish shared understandings or a common knowledge base". PTs are made by people with complementary skills who work together to achieve a common purpose for which they are accountable. They are enforced by their organization and follow specific strategies or organizational guidelines (e.g. time-to-market, effectiveness, low-cost, etc.). Their final goal is delivery of a product or service which responds to the requirements provided. Compared to the other OSS types, they are the most formal type of group. PTs are also defined as strict and single-minded aggregates of people, (closely) collaborating on well-defined reification tasks (i.e. tasks which produce a tangible artifact which justifies their effort). Any Scrum project team, e.g. in [20], is a project team.

Table I compares the OSS types. Column one, contains the attribute types that, according to literature, characterize OSSs. These attribute types occur often in literature describing with a specific value a specific OSS type. For instance: the "geodispersion" attribute type, has value "Network-Spanning" in NoPs, since the nodes part of a NoP are distanced in both time and space (i.e. every one has a different geolocation and their dispersion is network wide). Each OSS type is identified by a unique set of attributes' value which are typical for that specific type and not for others. Columns two to six, classify each type by giving its attributes their specific value. The remaining 8 types from our study don't comply with GSE definitions in [8], [7], [16] or [6] since they do not pursue goals that are typical for GSE projects. For example, Problem Solving Communities (PSCs) entail extremely experienced professionals to focus on solving a specific problem to achieve a strategic business advantage. PSCs are distant from GSE practice, because single GSE projects don't pursue the strategic advantage of organizational sponsors, but rather use the strategy advantage of round-the-clock productivity to deliver products faster. Other examples are Learning Communities and Strategic Communities, which are specific to practices around learning and development of best-practices. These communities are not intended to share a practice for a purpose (like in CoPs or NoPs), rather they are bent on pure learning. Lastly, types such as Social Networks and Informal Networks are too generic to be considered similar to a domain area such as GSE.

To understand which (sum) of these types matched GSE, we used empirical evidence from [4] and [15]. In these works, the authors present three GSE case studies, conducted within large GSE corporations, over a period of nine years. Through action research, 25 organizational factors were derived and refined, based on organizational problems and issues observed. These 25 factors are defined as key decisions that need to be taken for management and governance in GSE. Table II shows the 25 factors and exemplifies the decisions to be made for each factor. On Table II we made a distinction: the top part shows "socio-organizational factors" which refer to structure or operational behavior of teams (i.e. relating to people involved); the bottom part, in bold, shows "process management and efficiency factors" which refer to aspects of the software process or the efficiency of the GSE organizational structure (i.e. relating to the processes and approaches adopted).

B. Mapping OSS Types to GSE Factors

In the previous section we reported five OSS types which were similar (by definition) to GSE. To understand which combination of these five could effectively support GSE, we did a systematic concept mapping of the GSE organizational factors to OSS types. All 25 factors were mapped with attributes' values from the OSS types in Figure 2, and vice versa. More in particular, the following rule was applied:

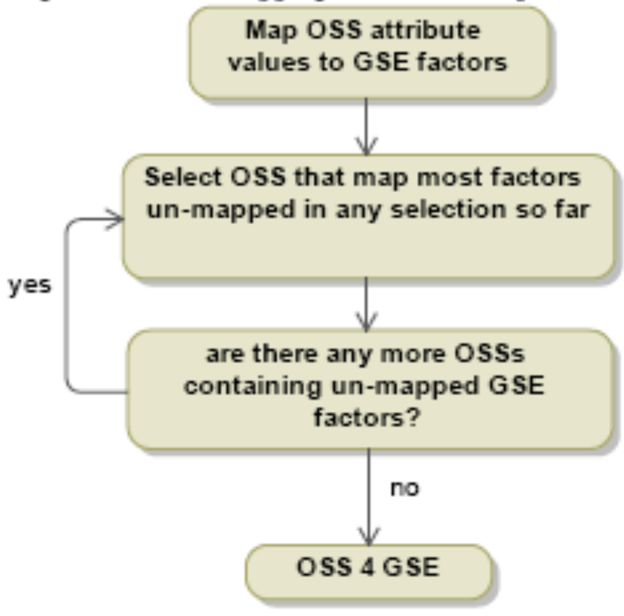
An OSS attribute "X" is mapped to a GSE organizational factor "Y", if and only if X's value is a decision about organizational factor Y

For example, the "Visibility" organizational factor was mapped to the attribute value "Visibility = Highest" from Knowledge Communities. Consequently, in this example, the Knowledge Community type is supportive to GSE, since it supports its "Visibility" organizational factor.

As a result of the mapping, we selected 4 types, namely: Project Teams, Networks of Practice, Knowledge Communities and Formal Groups. The "CoP – Community of Practice" OSS type was not selected since its organizational factors were previously mapped by other OSSs (which contained more factors). This mapping ensures that GSE organizational factors are mapped to supportive OSS attributes (and, consequently, to the OSSs to which they belong). Specifically, select OSS types consistently, we used a "greedy" rule, i.e. we selected a minimal set of OSSs by applying the following rule:

An OSS is selected if and only if its attributes' values, map to a set of organizational factors which were not present in previous OSS selections.

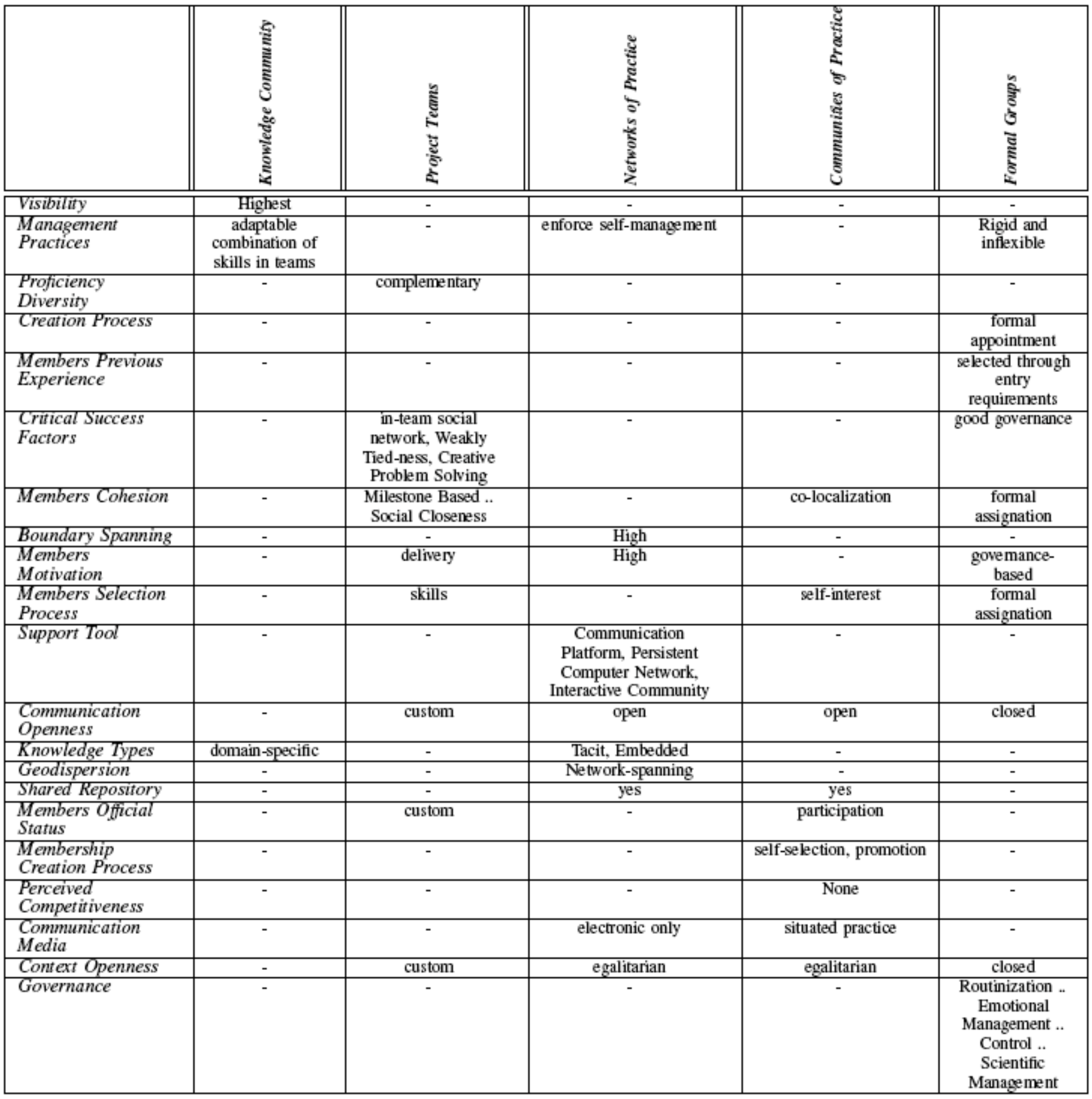
Figure 3. OSS mapping and Selection process


Map OSS attribute values to GSE factors
Select OSS that map most factors un-mapped in any selection so far
yes
are there any more OSSs containing un-mapped GSE factors?
no
OSS 4 GSE

The mapping process is summarized in the UML-style activity diagram in Figure 3.

The resulting OSS composite we selected, has two sets of attributes: (a) attributes that map GSE organizational factors; (b) other attributes of mapped OSS types. The composite OSS type for GSE can be described as follows: GSE practitioners are organized in project teams; also, they are forced to collaborate within a network (through the internet), within which they share a common practice (global software engineering) and the resources relevant to it (software artifacts being produced or used). Moreover, practitioners in GSE carry out knowledge intensive activities (i.e. preparing documentation, resolving requirement conflicts, making design decisions, etc.) across time and space distance, and this makes them similar to knowledge communities. Finally, their status is formally acknowledged (and governed) as (de-)centralized groups (e.g. development "sites"). Table IV (in the appendix) is a is a 19 x 25 matrix containing the OSS attributes' value / 25 GSE Organizational factors. Columns in bold are factors which remain unmatched by any attribute of any of the four OSS types we selected for GSE.

Every (I x J) cell matches the I-th attribute value (and consequently the OSS type to which it belongs) with the organizational factor in the J-th column.



	Knowledge Community	*Project Teams*	*Networks of Practice*	*Communities of Practice*	*Formal Groups*
Visibility	Highest	-	-	-	-
Management Practices	adaptable combination of skills in teams	-	enforce self-management	-	Rigid and inflexible
Proficiency Diversity	-	complementary	-	-	-
Creation Process	-	-	-	-	formal appointment
Members Previous Experience	-	-	-	-	selected through entry requirements
Critical Success Factors	-	in-team social network, Weakly Tied-ness, Creative Problem Solving	-	-	good governance
Members Cohesion	-	Milestone Based .. Social Closeness	-	co-localization	formal assignation
Boundary Spanning	-	-	High	-	-
Members Motivation	-	delivery	High	-	governance-based
Members Selection Process	-	skills	-	self-interest	formal assignation
Support Tool	-	-	Communication Platform, Persistent Computer Network, Interactive Community	-	-
Communication Openness	-	custom	open	open	closed
Knowledge Types	domain-specific	-	Tacit, Embedded	-	-
Geodispersion	-	-	Network-spanning	-	-
Shared Repository	-	-	yes	yes	-
Members Official Status	-	custom	-	participation	-
Membership Creation Process	-	-	-	self-selection, promotion	-
Perceived Competitiveness	-	-	-	None	-
Communication Media	-	-	electronic only	situated practice	-
Context Openness	-	custom	egalitarian	egalitarian	closed
Governance	-	-	-	-	Routinization .. Emotional Management .. Control .. Scientific Management

Table I: OSS COMPARISON TABLE

III. DISCUSSION

Two key observations were made on our results. First, 10 out of 25 organizational factors are not covered by any OSS (in bold on table II and table IV). These are “Project Management”, “Efficient Partitioning”, “Risk Management”, “Language Selection”, “Tools”, “Culture”, “Information”, “True Cost”, “Reporting” and “Process”. It is noticeable that all of the un-matched attributes are into the “process management and organizational efficiency” category (see section II-B), except “Culture” which can be a considered cross-cutting concern. This suggests OSS literature has not (fully) explored these factors yet. This should not come as a surprise, since OSSs focus on organizing people rather than supporting explicitly business purposes (i.e. intended to produce a tangible output such as software). This notwithstanding, we look at additional ramifications of the OSS meta-model resulting from our systematic literature review 2. We found that no combination of OSSs can offer support for all these factors: the “Strategic Community” type (a specialization of Communities of Practice) offers support to “Project Management”, “True Cost”, and “Fear” through organizational sponsoring practices, contract value management, as well as team partitioning guidelines (in the form of “previous experience” policies, “personal goals”). This suggests that the OSS type for GSE could be enriched by integrating features of a strategic community. Finally, the “Workgroup” type (which inherits from Networks of Practice) provides cohesion practices that could support both “Risk Management” and “Fear”.

Process management and organizational efficiency in GSE could benefit from support in the GSE OSS. Additional research should be invested in constructing an OSS hybrid which can cover all 25 organizational factors.

Factor	*Example*
Communication	Team A needs to communicate heavily with teams in location B given the dependency between their work packages. Live and efficient mechanisms for communication must be selected.
Communication Tools	Team A and others must agree on a tool that they all should adopt, in order to avoid miscommunication as much as possible
Temporal Issues	Team A and Team B are working on the same work page in two contiguous shifts of 8 hours. Information continuity should be planned.
Effective Partitioning	Team A should be entrusted with continuing the workpackage of B only if Team B can communicate efficiently with A
Skill Management	Team A has a large array of skills, Team B has a specific set of skills. Team A and Team B should not work on the same workpackage
Knowledge Transfer	Team A, B and C need to realize round-the-clock productivity with three consecutive, 8-hour shifts. Information continuity must be allowed and transfer of information needs to be planned.
Defined Roles / Responsibilities	Person "a" in team A is leader and made responsible for timely delivery. If "a" turns over, then person "a.1" should take over.
Team Selection	Task 1 will be carried out by team A, which is composed of people a+b+c; Task 2 will be outsourced to partner X; Task 3 will be carried out by team B from project Y.
Motivation	Manager 1 in site X adopts informal leadership approaches [1] to motivate Teams A and B while they work within a large GSE project.
Technical Support	Teams X and Y will be available 24/7 to provide network-wide support on workspaces, IDEs, codebase versioning and (in case of emergency) back-ups
Coordination	Team 1 will finish working on Task A at day D+11; Team 2 will wait day D+12 then make sure the release of task A is complete and start integrating Task A and B.
Cooperation	Teams 1 and 2 must pool resources on task A, since it requires their combined set of skills.
Teamness	Developer X of team A often leaves daily stand-up meetings beforehand or is uncooperative towards women in the same team, he cannot work properly in team A (50% women).
Visibility	Teams A and B should render their progress visible to all the development network since their timely delivery is critical for the good-health of the whole project.
Trust	After every stand-up meeting, developer X in off-shore team A phones the headquarters to verify instructions received; developer X doesn't trust leadership in team A.
Fear	developers X and Y are often heard talking about moving to India since their current site will be closed due to out-sourcing; as a result, the rest of the developers at their site are developing uncooperatively towards outsourcing attempts.
Project Management	**Deliverable A.1 is late of three days; deliverable A.2 depends on A.1 but is more critical. Deliverable A.2 should be started from partial results of A.1 and adjusted live.**
Effective Partitioning	**Teams A, B and C are in three different timezones. Only two of these timezones are contiguous in shift (Teams A and B). Teams A and B should work on the same work package. Team C should work on a work package as independent as possible to Team B's.**
Risk Management	**Technological Gateway at site A, is developer X. X is constitutes a single point of failure. Developer Y should be instructed to follow X and take-over as needed.**
Language Selection	**language at all sites should be homogenized to english. All sites should select english-certified developers so that international collaboration is possible.**
Tools	**technical space of each JAVA developer should be eclipse-centric; technical space of each designer and modeler should make extensive use of UML technologies and standards based on it.**
Information	**three types of information should be supported within the technical space of each developer at each site: models; documents; codebase. All information should be exchanged through secured emails and traceable.**
True Cost	**underlying costs (over-times, holiday, latencies and idle-times, etc.) must be calculated at each site and final figures should be summed up at every monthly meeting.**
Culture	**Teams 1 and 2 can work hand in hand since they are part of the same nation and are not limited by different national holidays or shift-times.**
Reporting	**BIRT should be available at every workstation; every developer should comment every artifact produced or retouched; all should be aware of their gateway (human or technological) towards other sites.**
Process	**Scrum sprints will be used to develop the project.**

Table II 25 ORGANIZATIONAL FACTORS IN GSE.

Second, Table III shows the composite OSS for GSE: column 1 contains the OSS types; column 2 contains the attributes' values, rephrased to represent the GSE domain; column 3 provides a label for unique identification; column 4 distinguishes new attributes from others that were previously explored in GSE literature (i.e. are matched by organizational factors). As expected, many of the attributes in Table III (e.g. attributes R2a and R2e) are not new to software engineering practice and, specifically, to GSE (e.g. [6], [3], [8]). This confirms that software engineering research has moved well in coping with many social and organizational factors occurring in GSE. On the other hand many attributes deriving from OSS characteristics, are new.

New attributes are marked with a capital "yes" on table III. Attribute R1a states this explicitly, by calling for dynamic indexing and retrieval of professionals (i.e. management must be aware of who is able to do what, and where). Attributes R2b, R2f, R3d, R3e, suggest ways in which people should be modelled or organized so that their skills can be easily retrieved or switched (i.e. people should be organized in such a way as to ease management's awareness of their abilities). Finally, attribute R3g, states a way in which people should be formatted in a federated social network to allow for their cooperation. At a first glance, from the description in table III, these new attributes seem to fall under the "skills management" factor. Rather, with the exception of R2i (which can be seen as a concern crosscutting all factors), they all address a different concern: awareness management (e.g. awareness of skills, awareness of the location and skills of certain people, awareness of task allocations, awareness of possible (re-)allocations of tasks to skills and people etc.).

OSSs	*attribute for GSE*	*Label*	*new?*
Knowledge Communities	the OSS_{GSE} type must support the application of management practices to index and dynamically retrieve skills from the professionals enrolled in it as needed (i.e. according to business demand);	R1a	YES
	the OSS_{GSE} type should have as prime goal knowledge generation and sharing;	R1b	
	the OSS_{GSE} type should adopt all possible ways to increase both internal and external visibility (e.g. local promotion, bannering, seminars, ad-hoc trainings, etc.);	R1c	
	the OSS_{GSE} type must envision ways to maintain its visibility at the highest level (e.g. by embedding itself with the technical space of the developers);	R1d	
Networks of Practice	the OSS_{GSE} type should ensure communication openness;	R2a	
	the OSS_{GSE} type should make explicit the geolocation (e.g. the location in both time and space on the globe) of each node;	R2b	YES
	the OSS_{GSE} type should support fine-grained skills management practices actionable on each node (e.g. it should be able to propose skill alternatives for each node);	R2c	
	the OSS_{GSE} type should clearly define which boundary objects (emails, blogs, RSS feeds, etc.) can be used and how (post-reply, knowledge repository, etc.);	R2d	
	the OSS_{GSE} type should provide a shared repository of knowledge to be maintained (automatically);	R2e	
	the OSS_{GSE} type should be able to use the shared knowledge repository to tighten the geodispersion of each node from the others (e.g. by using massive geo-coding technologies to locate each and every resource contained);	R2f	YES
	the OSS_{GSE} type should allow the application of governance practices (e.g. agreed norms, sanctions, automated rewarding mechanisms, emotional management, social events, etc.) on each network node to maintain its high motivation;	R2g	
	the OSS_{GSE} type should support integration with the technical space(s) decided for the project it is supporting;	R2h	
	the OSS_{GSE} type should allow the definition of (and agreement to) organizational practices (i.e. it should support the building of an organizational culture) based on company adopted standards and accepted values (e.g. as an entry pre-requisite to the OSS);	R2i	YES
Project Teams	the OSS_{GSE} type should support the guideline of "proficiency diversity = complementary" to support the definition of roles and responsibilities in project teams;	R3a	
	the OSS_{GSE} type should enable and nurture cohesion practices (e.g. proposal of team building initiatives, stand-up meetings, hang-outs, etc.) in project teams to maintain its high motivation;	R3b	
	the OSS_{GSE} type should integrate collaborative networking or programming facilities (e.g. CVS, distributed black-board, etc.);	R3c	
	the OSS_{GSE} type should allow the definition of an in-team technological gatekeeper, i.e. a person or entity which decides whom to forward certain technological-related issues or solutions;	R3d	YES
	the OSS_{GSE} type should assume each developer (i.e. each node) is weakly tied to the rest of the network, in case seamless switching of skills is needed;	R3e	YES
	the OSS_{GSE} type should nurture the creative problem solving abilities of project members (e.g. by integrating mind-mapping facilities);	R3f	
	the OSS_{GSE} type should allow the definition of a (federated) social network for local project teams;	R3g	YES
Formal Groups	the OSS_{GSE} type should integrate governance mechanisms for emotional management to mitigate fear and its negative potentials	R4a	
	the OSS_{GSE} type should integrate trust-in-members mechanisms (e.g. members trust-estimation) and practices (e.g. team-building);	R4b	

Table III OSS FOR GSE ATTRIBUTES

This trend we identified in the attributes indicates a new GSE factor focusing on awareness management in GSE. Using the new attributes, ad-hoc support tools could be developed, to support this new concern (e.g. an adaptable and dynamic social network of skills, rather than teams, to allow for their (re-)localization as needed).

IV. CONCLUSIONS AND FUTURE WORK

This paper provided a profile of the organizational social structure for GSE. Our data and discussions support two key conclusions.

First, current OSSs fail to support "process management and efficiency" factors in GSE. Additional investigation should be invested in devising an OSS which matches all 25 organizational factors relevant in GSE. For example, the definition of the complete OSS for GSE could be used to devise ad-hoc support tools to bootstrap GSE projects and monitor them.

Second, current practices in management and governance of GSE, have focused on the process, on coordinating organizations involved, organizing teams into coherent working units: additional effort should be invested in exploring mechanisms for awareness management in GSE. For example, mechanisms to support representation and (re-)localization of skills could be critical when certain project tasks remain dangling (e.g. as a consequence of employee turnover). Also, awareness should be supported at different granularity levels (i.e. skills, people, tasks, etc.).

In the future, we plan to develop a prototype to incrementally satisfy the profile of the OSS for GSE and the requirements it imposes. Future work should also be invested in developing a context-model of the OSS defined in table III (e.g. by refining the one presented in [18]), so that context awareness and adaptation mechanisms can be developed for the GSE OSS. This can be done by investigating further in the OSS state of the art (e.g. as provided in [17]) to identify attributes and characteristics which are part of OSS context.

ACKNOWLEDGMENT
The authors acknowledge the support of the European Community's Seventh Programme FP7/2007-2013, grant agreement 215483 (S-Cube), for partially funding this project.

REFERENCES

[1] Anne Bourhis, Line Dubac and Ral Jacob. The success of virtual communities of practice: The leadership factor. Electronic Journal of Knowledge Management, 3(1):23–34, jul 2005.
[2] John Seely Brown and Paul Duguid. Knowledge and organization: A social-practice perspective. Organization Science, 12(2):198–213, mar 2001.
[3] E. Carmel and R. Agarwal. Tactical approaches for alleviating distance in global software development. IEEE Software, 2(18):22–29, March/April 2001.
[4] Valentine Casey and Ita Richardson. Implementation of global software development: a structured approach. Software Process: Improvement and Practice, 14(5):247–262, 2009.
[5] Angela M. Dickinson. Knowledge sharing in cyberspace: Virtual knowledge communities. pages 457–471, 2002.
[6] Christof Ebert and Philip De Neve. Surviving global software development. IEEE Software, 18(2):62–69, 2001.
[7] James D. Herbsleb. Global software engineering: The future of socio-technical coordination. In Lionel C. Briand and Alexander L. Wolf, editors, FOSE, pages 188–198, 2007.
[8] James D. Herbsleb and Audris Mockus. An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering, 29(6):481–94, 2003.
[9] James D. Herbsleb and Deependra Moitra. Guest editors' introduction: Global software development. IEEE Software, 18:16–20, 2001.
[10] Eli Hustad. Managing structural diversity: the case of boundary spanning networks. Electronic Journal of Knowledge Management, 5(4):399–409, dec 2007.
[11] Eli Hustad. Exploring knowledge work practices and evolution in distributed networks of practice. Electronic Journal of Knowledge Management, 8(1):69–78, jan 2010.
[12] Michael Jones and Irit Alony. Guiding the use of grounded theory in doctoral studies an example from the Australian film industry. International Journal of Doctoral Studies, 2011.
[13] Lars Lindkvist. Knowledge communities and knowledge collectivities: A typology of knowledge work in groups. Journal of Management Studies, 42(6):1189–1210, sep 2005.
[14] Hartmut Raffler, Matthias Schneider-Hufschmidt, and Thomas Khme. System ergonomics and human-computer interaction at siemens corporate research and development. In Penny Bauersfeld, John Bennett, and Gene Lynch, editors, CHI, pages 65–66. ACM, 1992.

[15] Ita Richardson, Valentine Casey, John Burton, and Fergal McCaffery. Global software engineering: A software process approach. In Ivan Mistrik, J. Grundy, A. van der Hoek, and J. Whitehead, editors, Collaborative Software Engineering. Springer, January 2010.
[16] Raghvinder Sangwan, Matthew Bass, Neel Mullick, Daniel J. Paulish, and Juergen Kazmeier. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications, Boston, MA, USA, 2006.
[17] Damian A. Tamburri, Patricia Lago, and Hans Van Vliet. Organizational social structures systematic literature review. Work in progress, 2012.
[18] Damian Andrew Tamburri and Patricia Lago. Supporting communication and cooperation in global software development with agile service networks. In ECSA, pages 236–243, 2011.
[19] E. C. Wenger and W. M. Snyder. Communities of practice: The organizational frontier. Harvard Business Review, 78(1):139–+, January 2000.
[20] Laurie Williams, Gabe Brown, Adam Meltzer, and Nachiappan Nagappan. Scrum + engineering practices: Experiences of three microsoft teams. In ESEM, pages 463–471. IEEE, 2011.

	Communication	Communication Tools	Temporal Issues	Effective Partitioning	Project Management	Skill Management	Knowledge Transfer	Defined Roles / Responsibilities	Team Selection	Risk Management	Language	Motivation	Technical Support	Coordination	Cooperation	Tools	Culture	Teamness	Visibility	Information	Fear	Trust	True Cost	Reporting	Process
Knowledge Community																									
-Visibility = Highest																			x						
-Management Practices = Combination						x																			
Project Teams																									
-Proficiency Diversity = Complementary								x																	
-Creation Process = Formal Appointment									x																
-Critical Success Factor = Creative Problem Solving									x																
-Members Previous Experience = Cross Functional									x																
-Critical Success Factor = Weakly Tiedness									x																
-Critical Success Factor = Within Team Social Network												x													
-Members Cohesion = Milestone Based . Social Closeness								x				x						x							
Networks of Practice																									
-Boundary Spanning = High						x									x										
-Members Motivation = High						x						x													
-Members Selection Process = Entry Requirements						x																			
-Support Tool = Communication Platform, Persistent Computer Network, Interactive Community		x											x												
-Communication Openness = Open	x					x								x											
-Management Practices = Strong												x													
-Knowledge Type = Tacit							x																		
-Knowledge Type = Embedded							x																		
-Geodispersion = Whole Network			x																						
Formal Groups																									
-Governance = Routinization _ Emotional Management _ Control _ Scientific Management																					x	x			x

Table IV MATCHING OF OSS ATTRIBUTES TO GSE FACTORS.

Appendix H: A Survey of SOA Migration in Industry

Maryam Razavian and Patricia Lago

1 Introduction

Migration of legacy systems to service-based systems enables enterprises to achieve advantages offered by SOA, while reusing the business functions embedded in the legacy systems. Enterprises nowadays have many software systems that are needed to be modernized because they are difficult to change and they cannot cope with everlasting requirements changes. Service-enabling the legacy systems allows enterprises to modernize their pre-existing business functions as added-value services, and therefore achieve SOA promises such as agility and flexibility. Hence, identification of migration strategies for service engineering is critical for migration of legacies, and SOA adoption in industrial setting.

So far, many SOA migration approaches have been proposed in both industry and academia with the ultimate goal of adoption in practice. There is, however, considerable difference between SOA migration approaches defined in academia and those emerged in industry. For example, while scientific approaches mainly take a reverse engineering perspective, industrial practitioners developed best practices in forward engineering from requirements to SOA technologies, where legacy code is not transformed but used as a reference. This difference pinpoints a potential gap between theory and practice. One of the key causes of such a gap is that the approaches proposed in academia do not fully fit the main goals and needs of practice. To bridge this gap, it is necessary to understand the properties of migration approaches that are both feasible and beneficial for practice.

This paper provides deeper understanding of the types of migration approaches in industrial practice. To this end, we conducted an industrial inter- view survey in seven leading SOA solution provider companies. To the best of our knowledge, this is the first survey of this kind. With the objective of understanding the industrial migration approaches, we designed and executed the interviews. Each interview was analyzed considering the constituent conceptual elements of a migration process as proposed in [1], including the activities carried out, the available knowledge assets, and the overall organization of migration process. Furthermore, we looked for the best practices that companies have developed out of experience for successful legacy migration.

As a result we found that, in fact, all companies converge to the same, one, common SOA migration approach. This suggests that industrial migration approaches converge to a similar set of activities, process organization, and best practices, in other words, with experience enterprises mature toward a similar approach to SOA migration. In addition, we contrasted the industrial approaches with academic ones, which we identified from a previous Systematic Literature Review (SLR) on SOA migration [2]. Here we use the results of the SLR to discuss the differences and draw promising directions for industry-relevant research.

2 Results

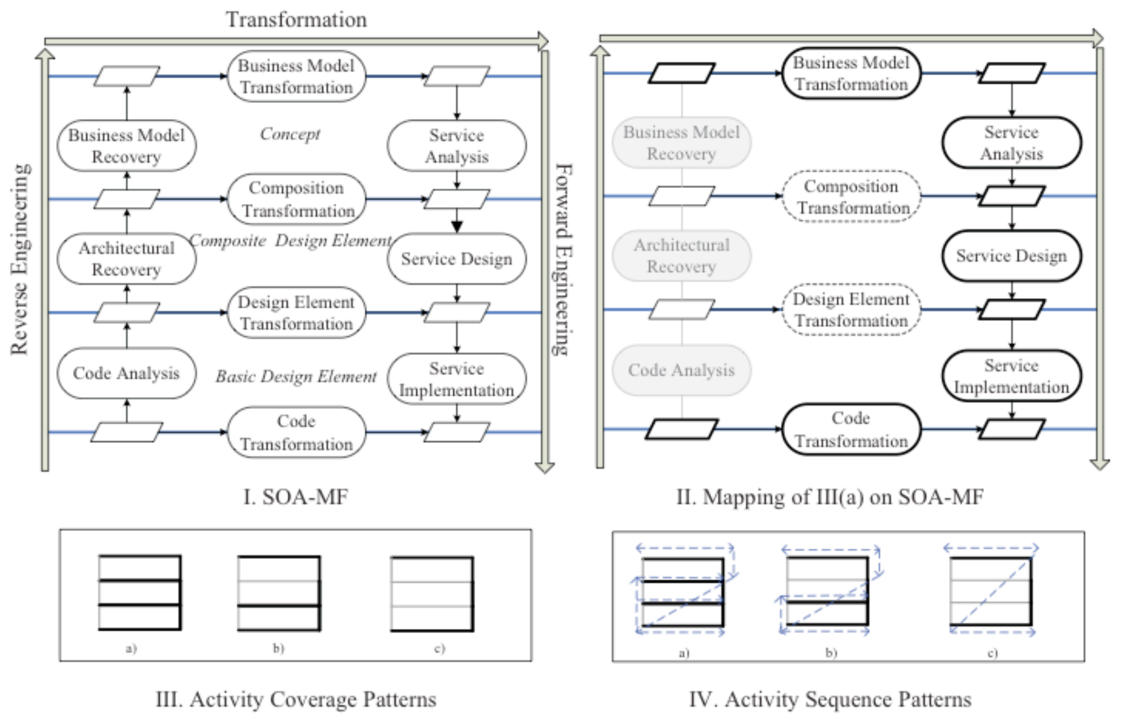
To gain an understanding on industrial migration approaches, we needed to typify the approaches in a unified manner. For this purpose, we used the SOA Migration Framework (SOA-MF) introduced in our earlier work [1] (see Fig. 1.I). The analysis of the approaches revealed patterns common among various com- panies1. These are listed in four key findings presented in this section. Each finding is summarized in a Reflection Box, which is followed by detailed discus- sion of the finding. Furthermore, each finding is compared with the results of our previous study on academic SOA migration approaches (the SLR mentioned in Section 1). Major differences between industrial approaches and academic ones can reflect gaps between theory and practice.

2.1 Migration Activities

Reflection Box.1.

- **F1.1.** Different companies share the same set of activities for migration.
- **F1.2.** Industrial migration approaches converge to one, common, type of migration.

To answer what is done in industrial approaches, we identified the constituent activities of various approaches and mapped them on SOA-MF. Fig. 1.III, represents the schematic forms of those mappings.



Mappings revealed two main findings: a) industrial approaches share the same set of activities for migration and b) industrial approaches are convergent to a subset of those activities. The two findings are further discussed in the following.

Finding F1.1. Various companies, independent from the company type (i.e., consultancy vs. in-house) and migration application domain, share the same set of activities for migration. This is evident from Fig. 1.III, where the activities correspond to three graphically similar coverage patterns. It should be noted that the similarity among coverage patterns, thanks to expressiveness of SOA- MF, indicates the conceptual similarity of constituent activities and artifacts of the migration approaches. According to [2], SOA migration approaches with similar set of activities constitute a migration family. Similarly, the three similar approaches identified in the interviewed companies belong to the same family. Contrast with theory. While the industrial approaches are all members of one family, the SLR revealed that the academic approaches belong to eight very different families. By covering different sets of activities each of these eight families provide a very different view on what SOA migration entails. For instance, one family reverse engineers the legacy code and transforms the extracted code segments to services, another family only covers the forward engineering sub- process. Considering the industrial approaches, all the approaches are categorized into (only) one of the eight families. Interestingly, the size of that family, called industrial family, is the smallest as compared to the others (i.e. 3% of academic approaches). Thus, one could conclude that 97% of the academic approaches do not fit in industrial family. This may indicate that academic research might be digging into aspects (like sub-processes and techniques) that are less relevant for industry. On the contrary, by looking at the characteristics of the industrial family research could better focus on the open research questions pertaining such family and hence have a better chance to close the gap between academic research results and industry needs.

Finding F1.2. By further analyzing the activities of the industrial approaches, we found that those common among all approaches, called core activities, are the ones shown in Fig. 1.II with bold boxes. The variable activities, i.e., those not common to all approaches, pertain to the coverage of the two transformation activities shown in Fig. 1.II by dashed line boxes. Furthermore, we observed that the core activities are those performed more frequently and systematically, while the variable activities are carried out less frequently and in an ad-hoc manner. More precisely, the limitations posed by legacy systems makes the variable activities less frequent. Several of the interview participants mentioned that, transformations that require decomposing the legacy systems are rarely carried out because they are not feasible as legacy systems are mainly monolithic. Furthermore, we observed that core activities are mainly supported by the state-of-the-practice methodologies and techniques such as SOMA [3]. The variable activities, however, are mainly carried out using local best practices. Consequently, we argue that, due to higher feasibility of the core activities and support of well-established methodologies and techniques, the industrial migration approaches are characterized by core activities.

Contrast with theory. None of the migration approaches in the SLR fully covers the core activities. I.e., none of the academic approaches comprehensively supports the type of migration that is both feasible and beneficial in indus- trial setting. This indicates an important gap between the migration activities emerged from practice and the ones researched in academia.

2.2 Sequencing of Migration Activities

Reflection Box.2.

– **F2.** In the industrial migration approaches the To-Be situation initiates and drives the migration.

By providing the mappings on SOA-MF, previous section addressed what activities are covered in the industrial migration approaches. Here we focus on what is the sequencing of those activities. There are two main types of sequencing of activities in the migration approaches, namely arc-shaped and bowl-shaped [4]. In summary, in arc-shaped approaches migration is driven by As-Is situation, while it is the To-Be situation that drives the bowl-shaped ones. All the industrial approaches elicited by our study were bowl-shaped.

This categorization of approaches is based on the graphical representation resulted from mapping their sequencing of activities on SOA-MF (e.g. Fig. 1.IV). The sequencing of activities in an arc-shaped approach starts from the reverse engineering sub-process. In this category, the As-Is situation initiates and drives the migration. Unlike the arc-shaped category, the bowl-shaped one starts from forward engineering and the To-Be situation is the main driver of migration.

Finding F2. The bowl-shaped sequencing of activities in industrial approaches implies the following: in all of the migration approaches the To-Be situation, characterized by requirements or properties of the target service-based system, drives and shapes the migration. To shape the migration process, first the To-Be situation is defined within the forward engineering sub-process; further, the To-Be situation is compared with the As-Is and as such, the legacy elements are selected and re-shaped to services. A question that arises is why industries perform migration in a bowl-shaped manner. Some of the participants, in one way or another, stated that in order to reach the migration goals they need to have the To-Be situation as the primary shaping force behind migration. As such, we conclude that to ensure achieving the migration goals, companies shape their migration decisions primarily by the To-Be situation.

Contrast with theory. Unlike the industrial migration approaches, the academic ones are mainly arc-shaped. In the SLR only 30 % of the primary studies are categorized as bowl-shaped approaches and the rest are arc-shaped. As such, 70% of the approaches do not support To-Be driven migration, which is considered as the best practice among the practitioners. This highlights promising opportunities for research to focus on how to support To-Be driven migration.

2.3 Legacy Understanding through Personalization

Reflection Box.3.

- **F3.1.** The industrial migration approaches do not use reverse-engineering techniques to understand the legacy systems.
- **F3.2.** The required knowledge is elicited from the stakeholders who own the knowledge.

Understanding the legacy systems plays an important role in SOA migration as it enhances extracting the best candidates among existing legacies for migration to SOA. In traditional software engineering, this understanding is gathered by extracting the representation of the legacy systems using reverse engineering techniques. As shown in Fig. 1.III, we observed that in the industry-defined approaches none covers the reverse-engineering subprocess. This observation resulted in two key findings discussed in the following.

Finding F3.1. To gain the required understanding of the legacy system, the industrial approaches do not use reverse engineering techniques. This is due to the following two reasons: a) the knowledge about the pre-existing system mainly resides in the stakeholders' minds (e.g. maintainer, developer, and architect). As such, the stakeholders know what functionalities are supported, and where they are located in the legacy system. As a result, reverse engineering of the pre-existing system is not favorable considering the little Return On Investment (ROI) it brings.

b) the legacy systems are usually comprised of a set of heterogeneous systems that are implemented in different programming languages ranging from COBOL to Java. As a result, for reverse engineering of the code different tools are needed and this implies a considerable amount of costs.

Contrast with theory. To understand the legacy systems, more than 60% of the approaches in the SLR use reverse engineering techniques. Those approaches extract the representations of the legacy systems using techniques such as code analysis and architectural recovery. Only one of the academic approaches (out of 39), supports the legacy understanding without reverse engineering techniques (i.e. using structured interviews)[5]. This indicates an important gap between theory and practice since reverse engineering is not favorable in practice.

Finding F3.2. We further observed that the industrial migration approaches elicit the relevant knowledge by directly asking the stakeholders, who own, developed, or maintained that system. More precisely, knowledge about the legacy system mainly remains tacit in stakeholders minds. As such, understanding is achieved by person-to-person knowledge elicitation. We argue that, this type of knowledge elicitation is in-line with personalization knowledge management strategy [6]. Personalization deals with exchanging tacit type of knowledge. Using personalization, the legacy understanding is gained by knowing 'who knows what' and consequently sharing the tacit knowledge about the legacy systems in that regard. Contrast with theory. In the SLR, all the approaches focus on capturing the knowledge by documenting it. As such, they are in-line with codification strategy addressing explicit documentation of the knowledge. The results of this study, however, suggests the importance of personalization. As such, research is needed to improve elicitation techniques, especially targeted for SOA migration, sup- porting personalization strategy.

2.4 Service Extraction by Defining the Ideal Services

Reflection Box.4.
- **F4.1.** The main driver in extraction of the legacy assets for migration is the portrait of ideal service.
- **F4.2.** Approaches emerged out of more experience portray the ideal services in more detail.

Finding F4.1. The migration approaches, inherently, embrace trade-off anal- ysis between the level of reuse of legacy elements and characteristics of the ideal services. We observed that, in this trade-off analysis, the industrial approaches assign considerably higher weight to the later rather than the former. To do so, first they determine the ideal services during the forward engineering sub- process. Later, those ideal services are re-shaped in a way that the reuse of pre-existing assets are realized. This way, the portrait of the ideal service is the main driver of service extraction. That is, the services identified from the pre-existing capabilities would likely be substantially different in the absence of that portrait of the ideal service. This is in-line with our other finding that all the migration approaches are bowl-shaped meaning that the To-Be candidate services guide the analysis and transformation of the As-Is legacy elements. Contrast with theory. A characteristic of the bowl-shaped approaches is hav- ing the ideal services (To-Be situation) as the main driver in service extraction. As such, this finding points out the same gap between theory and practice as discussed in finding F.2, namely inadequate support of To-Be driven migration.

Finding F4.2. We further observed that, industrial approaches vary in the level of detail in which they portray their ideal services. Some of the approaches only define the capability of the desired services at conceptual level (e.g. or- der business service), while some others also provide the design of such services along with its associated service contract (e.g. order software service design).

Some of the approaches externalize the constraints which each service should meet, while some others do not explicitly consider any constraints. Interestingly, we observed that the companies with more experience in providing service-based solutions tend to define the ideal services more detailed compared to the ones with less experience. Hence, we argue that the extent to which the ideal service is codified is an indicator of the maturity of the migration approach.

Contrast with theory. Detailed description of the ideal services is a best prac- tice that companies have developed with experience. Interestingly, we could not trace back this best practice to the academic approaches.

3 Discussion

In software engineering as an applied science, research in principle should serve the final purpose of being applied in practice. The extent to which this principle is supported by research, however, has been subject of debate for decades, and remains a still unsolved problem. The premier conference on software engineering featured in 2011 a panel on "What Industry Wants from Research" discussing the current gaps between theory and practice, and how to address them. All panel members in one way or another hinted the following cause of such gap: what research proposes does not fit the fundamental problems, goals, strategies and weaknesses of practice. We argue that, this paper is a step towards filling the theory and practice gap as it sheds light on how migration is performed in practice and further contrasting it with how academic research addressed the migration problem. By identifying the characteristics which make these approaches favorable for practice, we could identify directions for future research that have better chance of adoption by practitioners.

I) Migration approaches fitting core activities. Getting back to finding F1, we argue that core activities can act as a frame of mind confining the migration approaches that are more aligned with practice. From that perspective, one would see that, for instance, the approaches addressing wrapping the appli- cations as a whole are more in-line with practitioners concerns, compared to the ones addressing the automatic recovery of the legacy architecture. Hence, this frame of mind pinpoints the types of industry-relevant research in SOA migration methodologies and techniques.

II) To-Be driven migration approaches. As noted in finding F2, inadequate support for the bowl-shaped approaches in academia highlights promising opportunities for research to focus on how to support To-Be driven migration. For instance, future research can focus on addressing the following challenge of the practitioners: how to systematically elicit and capture the migration drivers and how to shape the migration process using those drivers.

III) Legacy understanding without reverse-engineering. Although re- verse engineering is not covered in industrial migration approaches (see finding F3), elicitation of the knowledge about the legacy system is crucial for a successful migration. In this regard, research can benefit practice by providing methods, techniques, or guidelines that facilitate elicitation of migration-relevant knowledge from different sources of such knowledge.

IV) Legacy evaluation from multiple perspectives. As noted, companies evaluate and extract the legacy assets for migration to SOA by depicting their ideal services. This is, however, done in an ad-hoc manner, which may hinder successful service extraction. An immediate concern calling for further research is how to systematically evaluate pre-existing legacy assets based on different aspects of the ideal services.

4 Conclusions

This paper explored the types of migration approaches employed by leading SOA solution providers in practice. Results show that by supporting similar set of activities, process organization, and best practices, industrial migration approaches do converge to one, common, type of migration. As such, this paper suggests that the industrial approaches mature towards a similar approach to SOA migration. Further findings (removed for sake of space) show that industrial approaches, strictly follow incremental migration.

In spite of what academics think, practitioners still face difficulties in con- solidating to a successful yet cost-effective migration approach. The many avail- able methods often prove to be abstract or commercial to be applicable. By contrasting the industrial migration approaches and the academic ones, this pa- per emphasizes important gaps between theory and practice and consequently sketches the promising industry-relevant research directions. Those research di- rections enable finding solutions to problems that industrial practice confronts in real-world migration cases and is tailored to individual needs.

When a company wants to devise or select a specific approach for migration of its pre-existing assets to services there are many issues that need to be resolved. In this study we identified the type of industrial migration approaches that is feasible in practice. What issues, goals, assumptions, and decisions explicitly make that specific type of migration favorable in practice, though, is yet unclear. We are carrying out follow-up studies to identify the goals, assumptions and issues that shape the migration decision making process.

References

1. Razavian, M., Lago, P.: Towards a Conceptual Framework for Legacy to SOA Migration. In: Fifth International Workshop on Engineering Service-Oriented Ap- plications (WESOA’09). (2010) 445–455

2. Razavian, M., Lago, P.: A Frame of Reference for SOA Migration. In: Towards a Service-Based Internet. Volume 6481 of Lecture Notes in Computer Science. (2010) 150–162

3. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. 47 (2008) 377–396

4. Razavian, M., Lago, P.: A Dashboard for SOA Migration. (2011) Under Submission. 5. Lewis, G., Smith, D.B.: Developing realistic approaches for the migration of legacy components to service-oriented architecture environments. In: Trends in enterprise

application architecture, Springer-Verlag (2007) 226–240 6. Hansen, M.T., Nohria, N., Tierney, T.: What’s your strategy for managing knowledge? Harvard Business Review 77(2) (1999)

Appendix I: The How and Why of SOA Migration in Industry

Maryam Razavian, Patricia Lago

Abstract—In industry, enterprises have many software systems to be modernized and made available as added-value services. The identification of migration strategies and practices for service engineering is critical for successful legacy migration, and SOA adoption in industrial setting. This paper presents the results of an interview survey on the migration strategies in industry. The purpose of this paper is twofold: 1) to discover the migration strategies that industrial practice adopts 2) to identify the uses of making such strategies explicit. Results of the survey have been analyzed with respect to migration activities, the available knowledge assets and the migration process. As a result we found that, in fact, all companies converge to the same, one, common SOA migration strategy. In addition, the uses of the strategy pinpoint promising industry- relevant research directions.

I. INTRODUCTION

Service-enabling the pre-existing legacy software is an important problem area in both research and practice. Enterprises have many software systems to be migrated and made available as added-value services. According to Gartner [1], enterprises migrate their legacy elements for three reasons: a) to retain legacy applications indefinitely due to their core position in the market, while coping with ever-changing requirements b) to improve business process efficiency and agility by integrating monolithic legacy systems c) to move to new delivery solutions such as software-as-a-service (SaaS). In order to achieve these important goals and to have an effective SOA adoption, identification of successful migration strategies is of great importance 1. Think of migration examples like the ABN- Amro/Fortis bank merger, involving thousands of software systems, thousands of applications in the IT portfolio of the two banks, and the data of millions of customers.

Many SOA migration approaches have been developed in both industry and academia [2]. Nevertheless, we have observed that the industrial approaches are considerably dif- ferent from the ones originated in academia. By discussing this observation with practitioners we were suggested that such differences might pinpoint an undesired gap between theory and practice. It is essential to fill this gap to devise solutions that fit the goals and problems of industry. This need was further emphasized most recently in a panel on “What Industry Wants from Research” in ICSE 2011 [3]. The general consensus among the panel members was that there is a need to better understand the fundamental problems, goals, strategies and weaknesses of practice. In this paper we provide such understanding by studying how migration is performed in industry.

To this end, we conducted an industrial interview sur- vey in seven leading SOA solution provider companies, followed by a panel of experts. Despite the diversity of participating enterprises, the interview survey revealed that they all converged to the same, one, common SOA migration strategy: all use similar input knowledge, similar activities, and sequences of activities to carry out migration. This suggests that with experience enterprises mature toward a similar migration approach. This would also confirm the SOA migration maturity level of Gartner Hype Cycle as being in early main stream phase [1]. In addition, and unlike the majority of academic approaches, SOA migration in industry mostly neglects reverse engineering. Rather, migration follows a forward engineering process initiated by identifying the ideal state (e.g. ideal business services), which is taken as a reference to extract and transform legacy elements to services.

The panel of experts following the interview survey in- vestigated the benefits of such overall strategy. The panel envisioned to use this strategy as a general tool to guide and steer migration projects. We further elicited a list of extensions to such tool, that would address the recurring problems in industrial migration, namely identification of the costs and risks of migration projects, and deciding on the best migration approach to mitigate them. The overall approach with extensions that emerges from the panel re- sembles the lean & mean approach of Kruchten [4], and draws interesting directions for industry-relevant research.

II. RELATED WORK

Since the early use of SOA, service-enabling legacy assets has caught a lot of attention. Various studies present an ap- proach for such migration [5], [6], [7]. All these approaches, aim for being adopted in practice. However, results show [8] that most of many academic migration approaches are not applied in practice, nor is their applicability in practice exemplified. This implies that those approaches have less chance of being adopted by practitioners. In addition, we identified very few works that discuss the migration ap- proaches emerged from practice [9], [10], [11]. Unfortu- nately, these approaches provide an ad-hoc representation specific for the particular case studied. As a result, it is not clear if they have the potential to provide a generally reusable holistic view on industrial migration approaches. By conducting a survey on migration approaches in various enterprises, this paper provides such holistic view. This work is to the best of our knowledge the first survey of its kind.

III. RESEARCH CONTEXT

In this paper we build upon our previous studies on SOA migration. Our research context is schematically depicted in Fig. 1, where the focus of this paper is given in white and earlier work in gray.

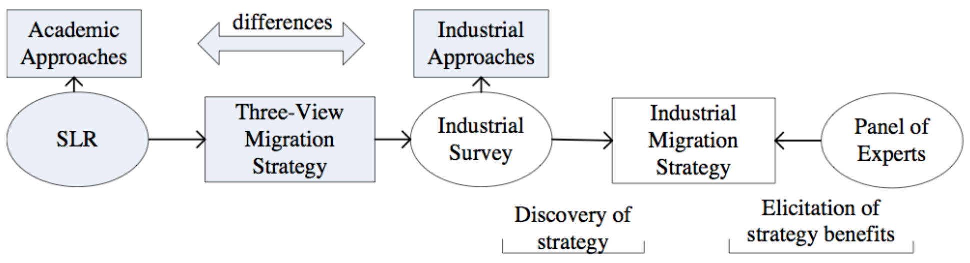


Figure 1. Research context

In our earlier work [2], we carried out a systematic literature review (SLR - see left-hand side of Fig. 1) on SOA migration in which we categorized Academic Approaches. As a co- product of SLR we further devised the theory of Three-View SOA Migration Strategy [8]. Section IV summarizes this theory. To gain an understanding on how industrial practice performs migration, we further conducted an Industrial survey in seven leading SOA solution provider companies. Results showed that the industrial migration approaches (In- dustrial Approaches in the figure) are considerably different from the academic ones [12].

These differences motivated us to seek for a deeper un- derstanding of industrial migration approaches. With this aim, we categorized the them using the three views. In this way we could uncover a common strategy emerged out of practice, called Industrial Migration Strategy which is presented in Section VI. It should be noted that, in this work we simply describe the strategy that resulted from the industrial survey. Accordingly, sections VI-B and VI-C partially recall information detailed in [12].

Finally, to verify the general applicability of the holistic industrial migration strategy and to understand its benefits we consulted a panel of experts. The results of the panel are reported in Section VII.

IV. BACKGROUND ON MIGRATION STRATEGY: A THREE VIEW PERSPECTIVE

Previous work showed that SOA migration can be effectively represented by three distinct yet inter-related views, each dealing with one aspect of a migration strategy:

(i) Activity view reflects what needs to be done in SOA migration, (ii) Knowledge view highlights the types of knowledge that shape and drive the migration, (iii) Sequence view focuses on the sequence in which the activities are carried out. We here discuss our rationale behind choosing this three-view representation.

a. Migration is a reengineering problem. We consider the problem of migration of legacy systems to SOA as a reengineering problem. In this view we follow a line of thought shared among various researchers. In [13] migration is defined as a modernization technique that moves the sys- tem to a new platform while retaining the original system's data and functionality. According to [14], reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent imple- mentation of the new form. The commonalities among these two definitions are considerable. In practice, the notions of "legacy migration", "integration" and "architectural recov- ery" which all deal with legacy applications, are considered as approaches to reengineering. b. Understanding As-Is and To-Be states. According to [15] any reengineering effort embraces a strategy indicat- ing how to move from the As-Is state to the To-Be state. As such, understanding the As-Is and To-Be states is essential for carrying out the migration. To determine the migration strategy, the concern of what knowledge elements define the As-Is and To-Be states have to be addressed. This concern is reflected in the knowledge view. c. Migration activities. To move from the As-Is to the To-Be state one needs to identify the best-fitting set of activities to perform the reengineering [15]. To this end, the decisions regarding the best-fitting activities have to be addressed in the migration strategy, that is the focus of activity view. d. Trade-offs between As-Is and To-Be states. Reenigeer- ing inherently embraces trade-off analysis between the de- gree of legacy leverage (i.e. As-Is state) and characteristics of the ideal state (i.e. To-Be state) [15]. As an example, the trade-offs between reuse of the monolithic and large legacy components, and well-granular services has to be frequently handled in migration projects. A migration strategy, should explicitly reflect concerns related to such trade-off analysis, namely, which of the As-Is and To-Be states has to be the key shaping force regarding the trade-offs as well as how the sequencing of activities should instrument the key shaping force. This concern is addressed in the sequence view.

Fig. 2 gives an example of three-view knowledge representation. The knowledge view (Fig. 2.I) indicates the available input knowledge with color green, and the required input knowledge that is not available with red. The knowl- edge conversions are represented using arrows. The activity view (Fig. 2.II) represents the activities covered in migration process. To do so this view maps the migration activities on the SOA Migration Framework (SOA-MF) introduced in our earlier work [16]. SOA-MF is a skeleton of the holistic migration process along with the distinct conceptual ele- ments involved in such a process. The framework consists of three sub-processes: reverse engineering, transformation and forward engineering. Finally the sequence view (Fig. 2.III) shows the order in which the covered activities are carried out, this being of two types: arc-shaped (because activities are carried out from left to right, driven by the As-Is state represented by the legacy code, i.e., resembling an arc) or bowl-shaped (because activities are ordered from right to left, driven by the To-Be state, resembling a bowl). By selecting one of the two types (see radio button in the figure) one indicates the overall ordering of activities.

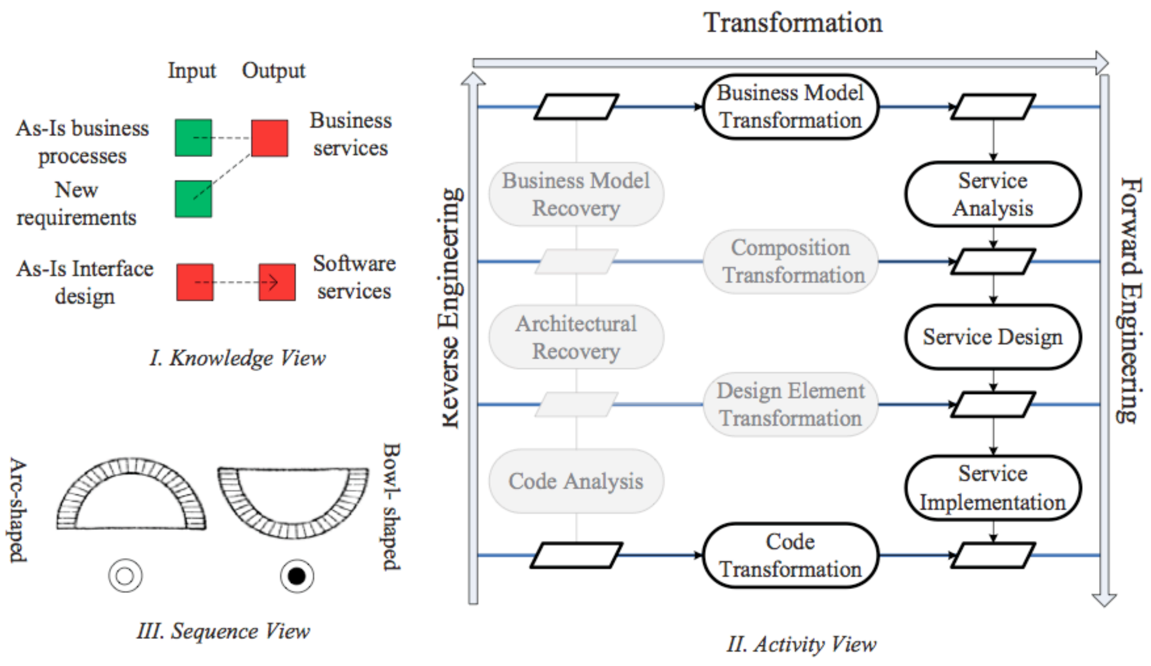


Figure 2. Three-view Strategy Representation

V. DESIGN OF INTERVIEW SURVEY STUDY

This section describes the research methodology of the study that is the focus of this paper (white area of Fig. 1). Here, we introduce the research questions, the study design, and data analysis method behind our industrial survey.

A. Research Questions

In order to investigate how migration is performed in industrial practice the following main research question was formulated: What approaches regarding legacy to SOA migration are used in practice? This main question is refined as:

- (RQ1) what are the knowledge elements that are used and produced?
- (RQ2) what are the activities carried out?
- (RQ3) what is the sequencing of the activities?

B. Study Design

We chose interview survey as our research method for two main reasons: (i) Survey is the favorable method for the type of questions our research seeks to answer. In Yin's categorization [17] our research questions fall under "what" category (as we look for what activities, what process and what good practices). These questions are likely to favor surveys, according to [17]. (ii) Our research is of explorative nature, hence requiring multiple data points. Surveys again are a suitable method for this purpose as they address a larger target population compared to other research methods such as case studies. To gather information about industrial SOA migration approaches, we conducted a series of interviews. Interviews are an appropriate strategy when the goal is to identify the experience of individuals and/or organizations in carrying out a task [18]. The conducted interviews were semi-structured. The open-ended questions of this type of interviews allow interviewers to ask follow-on questions when necessary. The first version of the interview guide was piloted with one researcher and one practitioner with experience in SOA migration 2. The theoretical population [17] target of this study included architects (both technical and enterprise architects) with considerable experience in carrying out SOA migration projects. To recruit suitable architects we created a leaflet illustrating the goals of this study. We distributed it both among our industrial network and in the Dutch architecture conference 2009, a well-known practitioner conference with over 500 participants each year. To ensure the reliability of the result we carefully chose the companies that are leading SOA solution providers. To ensure that the industrial survey is based on real-world experience, instead of participant's opinion on how migration should be carried out, we asked them to select one recent project and answer the questions considering that specific project. Finally nine architects, affiliated with seven international companies residing in the Netherlands and Belgium committed to our study. Table. I provides an overview of the information about the interviewees, their company and the domain of the migration project selected for each interview.

Before carrying out the interviews, the interviewees were sent a copy of the interview guide as well as some back- ground information on the study. This allowed us to better synchronize terminology. The interviews were conducted on site and were all video-recorded. The initial findings along with any remaining unanswered questions were iterated with the interviewees to reassure their correctness and completeness.

Interviewee code	Interviewee role	Company ID	Company Type	Company Size	Experience with SOA	Project Application Domain
RVS	Ent. Arch	A	in-house	82,500	11 yrs.	Telecom
CXB	Tech. Arch	B	Consultancy	50	9 yrs.	Public Ad-ministration
SVD	Ent. Arch	C	Consultancy	39,000	11 yrs.	Call Centers
JXM	Tech. Arch	D	in-house	10,000	10 yrs.	Banking and Insurance
PXB	Ent. Arch	D	in-house	10,000	10 yrs.	Banking and Insurance
DXL	Tech. Arch	E	Consultancy	388,000	11 yrs.	Utility and Energy
GWH	Ent. Arch	E	Consultancy	388,000	11 yrs.	Utility and Energy
RXB	Ent. Arch	F	Consultancy	350	8 yrs.	Finance
JWV	Ent. Arch	G	Consultancy	10	3 yrs.	Finance-Payroll

Table 1. Interviews overviews

C. Data Analysis

To typify the industrial migration approaches, we analyzed each of interview transcripts. We chose coding as our qualitative analysis method. In order to carry out the analysis systematically, inspired by the method of Miles and Huberman [19], we devised the following coding procedure for our purpose:

1) Surfing knowledge elements: Identifying the knowledge elements as well as the conversions among them (RQ1).

2) Filling in/Surfacing activities involved in migration: coding activities and refining the codes labeling activities, identifying the new activities (RQ2).

3) Filling in/Surfacing sequencing of activities: coding the inputs and outputs of an activity, identifying the sequence of activities (RQ3).

VI. RESULTS OF INDUSTRIAL SURVEY

In this section, we present the results of our analysis on the conducted interviews. Using the coding procedure explained in V-C, for each interview we codified the migration strategy used at its associated enterprise. Despite the many differences between the participating enterprises, the analysis revealed a great deal of similarity between their industrial migration strategies. The study, interestingly, shows that the approaches converge to, one, common strategy. More precisely, industrial migration strategies use and produce similar knowledge elements (knowledge view), perform similar activities (activity view), and follow the same sequencing for those activities (sequence view). Although we did not specifically ask if this common approach is successful, we see that this approach, which is working in practice, has emerged out of experience. In other words, with experience enterprises have matured toward a similar approach to SOA migration. In the reminder of this section we will describe the details of the analysis results with respect to each of the three views.

A. Knowledge View

By mapping each of the approaches on the knowledge view we isolated (i) the knowledge elements that enable the migration, and (ii) the conversions among those elements. The analysis of the mappings revealed that the knowledge view of the industrial approaches converge to the view shown in Fig. 3.I, and its implications are explained in the following.

The approaches share three main knowledge outputs, and also use similar input knowledge elements for creation of each output. These three main knowledge outputs are (see Fig. 3.I): (i) migration lifecycle plan, (ii) representation of ideal services, and (iii) software service models. In more details, for each of them we observed the following commonalities:

A Migration lifecycle plan - to decide on the migration increments. As reported in [12] migration processes are incremental. Accordingly, we observed that, similar to any iterative and incremental software process, migration processes also start with lifecycle planning. In this plan important decisions such as the number of increments, or the ordering of migrating pre-existing assets have to be reflected. As shown in Fig. 3.I, the knowledge elements that shape those targets are business goals (e.g. achieving agility), risks, pre-existing legacy assets, and constraints (e.g. time). Interestingly we found that to support these decisions, some of the surveyed companies have developed a number of local practices such as highest-value-service-first, easiest-service- first, and selection-using-enterprise-architecture. Using the first two practices, services with highest value for the market or services that are easiest and consequently fastest to create, are first extracted and migrated to SOA. The selection-using-enterprise-architecture practice suggests forming the sequencing of migrating services using business and information architecture of the regarding domain. As such, they can select independent increments to migrate (i.e., portions of legacy elements that can be migrated independently of each other), the sequencing of the increments to achieve the desired goal, and dealing with inter-dependencies among legacy elements.

Enterprise architecture - to understand As-Is and To- Be states. We further found that prior to identification of software services (to be migrated or newly made), identification of ideal services is carried out. To do so, the industrial approaches first achieve a high level understanding of the As-Is and To-Be states. An interesting observation here was that to gain such understanding all enterprises extract the enterprise architecture (EA) of both As-Is and To-Be states. As such, the industrial approaches (partially) capture the As- Is and To-Be states in terms of EA elements, i.e. business architecture, data architecture and technology architecture. Business architecture represents the structural and behavioral architecture of the business. The first represents the key business capabilities and the interrelationships among them. Examples of business architecture are legacy functional blueprint (CXB), business service pool (RVS) and domain architecture (RXB). The business behavioral architecture represents the main behavior of the business domain in terms of business processes and business rules. Data architecture represents the data entities representing the business. Finally, technology architecture articulates the embodied software, middleware and hardware technologies used in As-Is and To-Be states. Besides the As-Is and To-Be enterprise architecture, and to identify the ideal services, practitioners use as inputs new requirements and goals, and service-specific characteristics (e.g. loosely coupled services).

Ideal services - as main drivers. Having identified the ideal services, the next step is to extract the software service model, i.e., the actual services. We found the ideal service representation to be the main driver in this process. As noted, migration approaches inherently embrace trade-off analysis between the degree of reuse of legacy elements and characteristics of the ideal services. We observed that, in this trade-off analysis, the industrial approaches assign considerably higher weight to ideal services than to legacy reuse. To do so they reshape the ideal services in a way that the reuse of pre-existing assets are realized. This way, the representation of the ideal services as well as knowledge about the design of legacy systems are the main knowledge inputs for service extraction. This is evident from Fig. 3.I where the input knowledge elements are representation of ideal services as well as design-related knowledge of the legacy systems (i.e. design models, design constraints, and both desired and undesired qualities).

B. Activity View

To identify the constituent activities of industrial approaches, we mapped them on SOA-MF [2]. Fig. 3.II, represents the schematic forms of those mappings. Mappings revealed the following findings a) industrial approaches share the same set of activities for migration and b) industrial approaches are convergent to a subset of those activities. The two findings are further discussed in the following.

Various companies, regardless of their company type (i.e., consultancy vs. in-house) and market segment (e.g. telecom, banking, energy), share the same set of activities for migration. This is evident from Fig. 3.II, where the activities correspond to three graphically similar coverage patterns. By further analyzing the activities of the industrial approaches, we found that those common among all approaches, called core activities, are the ones shown in Fig. 3.III with bold boxes. The variable activities, i.e., those not common to all approaches, pertain to the coverage of the two transformation activities shown in Fig. 3.III by dashed line boxes. Furthermore, we observed that the core activities are those performed more frequently and systematically, while the variable activities are carried out less frequently and in an ad-hoc manner. Consequently, we found that, the industrial migration approaches converge to one, common, activity view which covers the core activities. The activity view, as such, represents the following commonalities among the migration approaches.

Core activities. Next to some activities that vary due to the specific domain or context of a migration project, we found that in all participating companies migration entails three core activities: a) gap analysis at EA level, b) forward engineering, and c) legacy application wrapping. More precisely, by covering business model transformation, the industrial approaches support gap analysis between As- Is and To-Be states that are illustrated using EA. Second, by covering the entire forward engineering process, the industrial migration approaches cover (not surprisingly) the three activities of service analysis, service design, and service implementation. Third, by covering the code trans- formation activity, all elicited migration approaches include transforming pre-existing applications as a whole to new target services. Transformation here, entails wrapping the legacy system without decomposing it.

Legacy understanding without reverse engineering. As shown in Fig. 3.III the reverse engineering sub-process is not covered in industrial migration. To gain the required understanding of the legacy system, the industrial approaches do not use reverse engineering techniques. This is because the knowledge about the pre-existing system mainly resides in the stakeholders' minds (e.g. maintainer, developer, and architect). As such, the stakeholders know what functionalities are supported, and where they are located in the legacy system. As a result, reverse engineering of the preexisting system is not favorable considering the little Return On Investment (ROI) it brings. We further observed that the industrial migration approaches elicit the relevant knowledge by directly asking the stakeholders, who own, developed, or maintained that system. More precisely, knowledge about the legacy system mainly remains tacit in stakeholders minds. As such, understanding is achieved by person-to-person knowledge elicitation.

C. Sequence View

The previous section described what activities are covered in the industrial migration approaches. Here we focus on the order in which the covered activities (e.g. Fig. 3.II) are carried out (e.g. Fig. 3.V). As mentioned in Section IV there are two types of sequencing: arc-shaped and bowl-shaped. The sequencing of activities in an arc-shaped approach starts from the reverse engineering sub-process. In this category, the As-Is state initiates and drives the migration. Unlike the arc-shaped category, the bowl-shaped starts from forward engineering and has the To-Be state as the main driver of migration.

Bowl-shaped sequencing. All the industrial approaches elicited by our study were bowl-shaped meaning that the To-Be state, characterized by requirements or properties of the target service-based system, drives and shapes the migration. A question that arises is why industries perform migration in a bowl-shaped manner. Some of the participants stated that in order to reach the migration goals they need to have the To-Be situation as the primary shaping force behind migration. For instance, CXB said: “We start the migration by defining the target blueprint (instead of identifying what are the pre-existing capabilities), otherwise we cannot ensure achieving the level of flexibility we envision”.

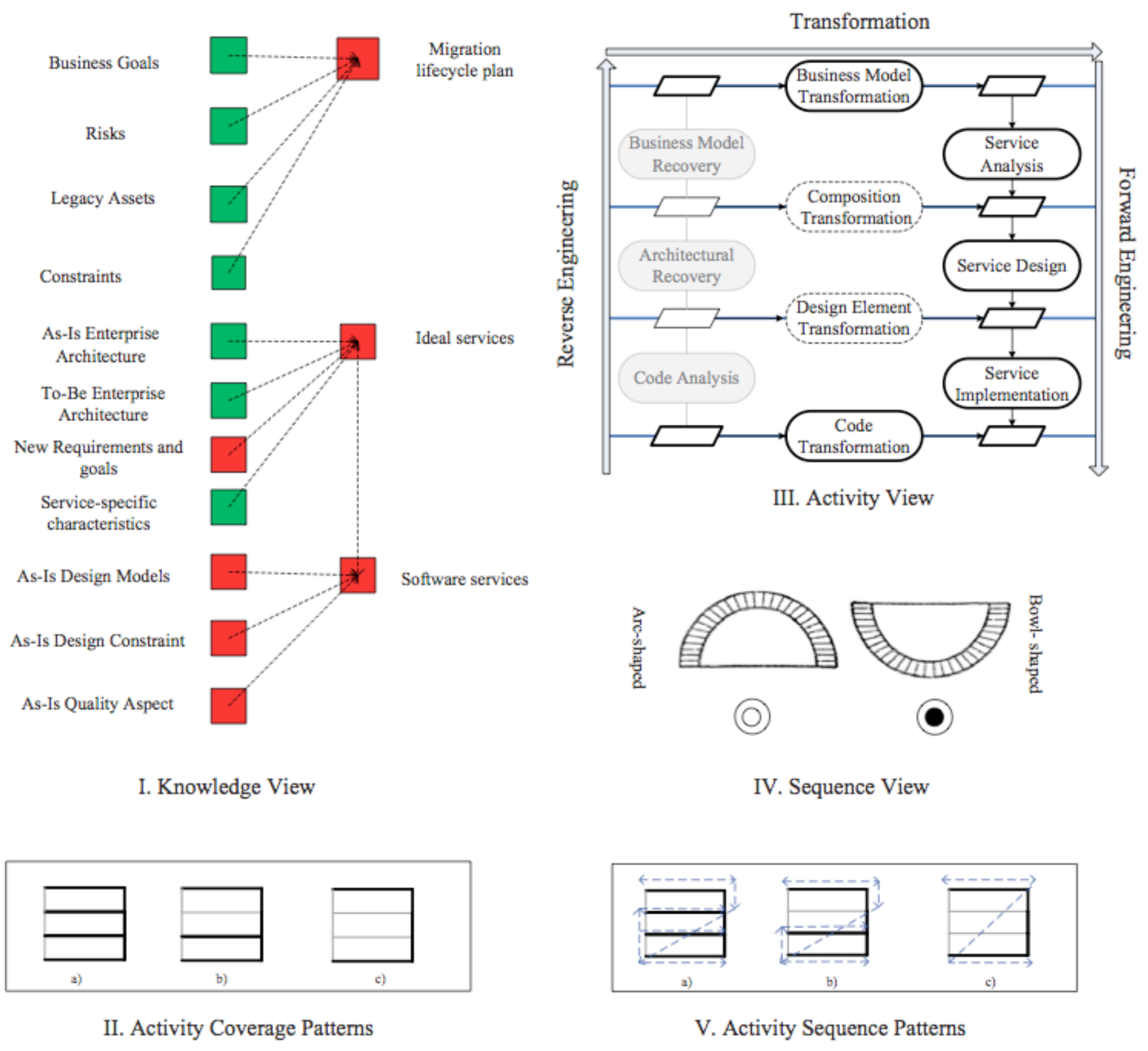


Figure 3. Three view Strategy Representation.

VII. A LEAN & MEAN MIGRATION STRATEGY

The industrial survey presented so far provides a general tool to represent and steer migration. This tool, in the form of the three-view migration strategy, is common to all seven participating companies. Whereas this result clearly helps giving structure to a migration project, we wanted to understand if formalizing it would help industry in solving important problems. In other words, would the three-view migration strategy provide concrete benefits? And what benefits?

To further dig into this question, we organized a panel of experts. We invited four senior architects from three different companies other than those seven participating in the industrial survey. We intentionally chose senior architects as they are the stakeholders who are aware of the key characteristics of the migration projects, have long-lasting experience in multiple projects and know the trends and current practice in use in the company. As a co-product of the panel we gathered further evidence on the commonality of the three-view migration strategy. Most importantly we elicited a very interesting list of concrete benefits that the three views could offer for their own company.

The benefits are discussed in the following. Overall, we observed that the way the panel of experts (or the panel in short) sees the three-views of our migration strategy resembles the idea of a lean & mean process model [4]. In his paper, Kruchten observed that the process models developed in the last decades are too rich, hence hindering process support rather than providing guidance. He argues that more meaningful (i.e. mean) and much simpler (i.e. lean) models would be better and wider applicable, and they could be extended and customized only in case of need. In a similar vein, our three-view migration strategy is both mean (covering what really matters for migration) and lean (screening out details specific to the project at hand). Interestingly enough, the benefits emerged during the panel discussion identify a whole set of extensions that would eventually offer customizations reusable by companies only if they need them.

For each view of our strategy, the following discusses the extensions elicited from the panel.

A. Knowledge View

Knowledge-view checklist. By specifying the main input and output knowledge elements, the knowledge view in practice acts as a checklist. The panel stated that the most important use of this checklist is not to identify what knowledge is available, but to focus on what is not available. In this way, practitioners can analyze early enough the costs and risks of eliciting the missing knowledge and eventually decide whether it is cost-effective to elicit the knowledge or not.

Extensions. The panel also identified various extensions to the knowledge view that provides benefits to migration. In the following, we describe the most important of those extensions and their related benefits.

1) What is the source of knowledge: documents or peo- ple? The panel found that the knowledge view should make the sources of knowledge explicit. The problem is that most of the times knowledge is not written in documents, but is kept in people’s mind. A major con- tribution of knowledge view in this case is to highlight where a knowledge resides (documents or people’s mind). Knowing this is essential for planning the right activities for knowledge elicitation. In this way, again, practitioners can better analyze and manage the costs and risks of knowledge elicitation activities. It should be noted that, the distinction between knowledge in documents and knowledge in people’s mind confirms Nonaka and Takeuchi’s two modes of knowledge: explicit and tacit [20].

2) Validity period for knowledge elements. In addition to the previous point, the panel observed that the knowl- edge view should make explicit if the availability of some knowledge elements is temporal, i.e. if it has an expiration time. Quite often migration projects last a long period of time, and people that participate in the beginning of the project leave the company or retire before it is finished. As people are typically important sources of knowledge, and knowing if they will become unavailable before project completion is essential to ensure knowledge transfer in time, hence avoiding delays and economic losses.

3) What can change. One interesting observation made by panel was that in the lifetime of a project some knowledge may change. The issue is that in some cases changes are frequent and have important impacts on migration. As such, if not planned those changes can result in extra costs and efforts. The panel commented that the knowledge view should be an instrument to identify and highlight which knowledge elements might undergo changes. In this way, practitioners can analyze early enough what can change and how to mitigate the related risks.

B. Activity View

Core activities. The panel found the activity patterns (Fig. 3.II) expressive and conforming to different categories of their migration projects. What was indicated as most useful was to set one of the activity patterns as the reference model of their migration process and further derive their process model using this reference. They also put forward a set of extensions for improving the activity view:

Extensions. The experts emphasized the importance of making explicit various concerns about the activities the most important being:

1) Costs and risks of activities. The panel emphasized that what is especially important for them is to explic- itly know the costs and risks of each migration activity. Associating costs and risks to activities makes the activity view an even more powerful tool for planning how to do migration. For instance, by knowing the costs of automatic reverse engineering, one might decide to gain understanding about the legacy system using alternative techniques (e.g. asking stakeholders).

2) Practices related to different activities. The panel also suggested to link the activities in the activity view to practices. This helps practitioners to select the prac- tices that are emerged out of experience and are proved to be beneficial for carrying out specific activities. This will result in saving considerable amount of time and costs. In summary, this extension would help to bring order to existing practices and further facilitating their selection, and reuse in similar situations.

C. Sequence View

Bowl-shaped sequencing. As for the sequence view, the panel unanimously confirmed that migration in practice typ- ically follows a bowl-shaped approach. Accordingly, the To- Be state drives the migration project, and the related goals should be used to regularly measure progress. The panel, however, commented that during a project practitioners loose track of such goals, eventually causing deviations and delays. While the sequence view is trivial and as such does not seem to bring specific benefits, labeling the migration projects with being “bowl-shaped” would increase awareness in project members about the goals related to the To-Be state and build suitable mechanisms to e.g. schedule assessments of the project progress, carry out such assessments, and make sure that a project remains on track.

Extensions. Notwithstanding SOA migration projects are overall bowl-shaped, the panel recognizes that the individual increments of the iterative incremental approach can be arc- shaped, too. For instance, while mergers typically migrate their information systems to achieve a uniform To-Be state, they also need to incrementally reverse engineer the data or the applications of the legacy systems of the merged com- panies to migrate to the new technology. This requires some increments to be arc-shaped. The panel indicated as very beneficial if such bowl- and arc-shaped increments would be made explicit in reusable patterns, and if they would be associated with typical risks, costs and pre-requisites. This extension of an overall bowl-shaped migration strategy with reusable arc- and bowl-shaped increments resembles the idea of a ”lean & mean” approach. The panel suggestion specifically sees as added value the ability to decide on the best extension based on prerequisites, risk- and cost assessments.

VIII. THREATS TO VALIDITY

Below, we discuss the validity threats regarding reliability and generalizability of this work and what we did to address them.

Internal validity. Internal validity aims at ensuring that the collected data enables the researchers to draw valid conclusions [21]. In this survey, the interviews are mainly conducted by a single researcher and hence subjective inter- pretations might exist. To mitigate this threat, the interview guide was checked and validated by senior researchers experienced in software engineering, empirical studies and SOA. Moreover, the first two interviews were coached by a professional consultant expert in the field of ‘interviewing in qualitative research’, followed by two reflection sessions to review the execution of the interviews. Threat to validity of the analysis is in the general applica- bility of the codes used for characterizing and classifying migration approaches. An assuring factor in this regard is that the start-list of codes is extracted from a conceptual framework published in a service-oriented computing forum, after being peer reviewed by experts in the field [16]. This framework stems from existing theory on reengineering and architectural recovery while it is constantly refined through our coding procedure. This further consolidates its general applicability. Finally to assure accuracy of findings, the three-view rep- resentation of approaches and initial findings were iterated with the interviewees to be confirmed.

External validity. External validity defines to what extent findings from the study can be generalized [21]. In this regard, a possible threat is that the survey is relatively small and the companies are mainly situated in the Netherlands and Belgium. As such, the architecture culture of Dutch enterprises might have influenced the results to be "architec- ture centric". To mitigate this threat, the interviewees were chosen from international companies that are geographically distributed. Nevertheless, as a follow-up study we plan to geographically extend the survey to further investigate the generalizability of results.

Moreover, to increase generalizabality we intentionally chose senior architects as they are the stakeholders who are aware of the key characteristics of the migration projects, have long-lasting experience in multiple projects and know the trends and current practice in use in the company. In order to cover different but relevant perspectives on the subject matter, we chose both enterprise and technical architects as interviewees.

Finally, we organized a panel with experts other than the ones participated in our study, over a year after the com- pletion of interviews. This further consolidates the general applicability of the results to and across populations of persons, and time.

IX. CONCLUSIONS

This paper presents the results of an interview survey on SOA migration strategies in industry. Our main research question was 'what approaches regarding SOA migration are used in practice?'. To answer it we first carried out the industrial survey with the goal of discovering the mi- gration strategies that industrial practice adopts. Second, we organized the panel of experts with the goal of identifying the uses (or industrial benefits) of making such migration strategies explicit.

Related to our first goal, the results of the survey show that despite the diversity of enterprises participating in the study and of their market position, their migration converges to one, common, strategy - driven by common types of knowledge elements and core activities, fundamentally bowl-shaped, and with little to no attention to reverse engineering. In addressing the second goal, the panel envisioned to use this converging migration strategy as a general tool to guide and steer migration projects. The overall approach with extensions that emerges from the panel resembles Kruchten's lean & mean approach, and draws the following promising directions for industry-relevant research:

(i) Aligning risks, costs and value with migration strategy: One of the issues that was repeatedly stated by the panel was the importance of risks and cost management in SOA migration decision making. We found risks and cost man- agement to be in fact one of the main drivers of migration and influential on most decisions. This further confirms a recent interest toward risk-, cost- and value-aware methods [22], [23] that needs further research.

(ii) Providing decision making tools to support selection of migration solutions: In addition to the previous point, panel emphasized that industry needs tools to support plan- ning and decision making for migration. For this purpose, the panel indicated as very beneficial to associate the practices or extensions with typical risks, costs and pre-requisites. This calls for empirical research studies to identify and isolate practices and associate them with important decision criteria such as risks, costs and pre-requisites.

These research directions enable finding solutions to problems that industrial practice confronts in real-world migration cases and is tailored to individual needs. As such those migration approaches can communicate better with practitioners and consequently better fill the gap between theory and practice.

ACKNOWLEDGMENTS

This research has received funding from Jacquard (con- tract 638.001.206 SAPIENSA: Service-enAbling PreexIsting ENterprISe Assets); and FP7 contract 215483 (S-Cube). We would like to thank all architects that participated in this study. Special thanks go to Eoin Woods and Hans van Vliet for their feedback on earlier versions of this paper.

REFERENCES

[1] G. Inc., “Hype Cycle for Application Development,” Tech. Rep., 2011.

[2] M. Razavian and P. Lago, “A Frame of Reference for SOA Migration,” in Towards a Service-Based Internet, ser. Lecture Notes in Computer Science, vol. 6481, 2010, pp. 150–162.

[3] “ICSE 2011 Panel on What Industry Wants from Research.” [Online]. Available: http://margaretannestorey. wordpress.com/2011/08/05/icse-2011-panel-on-%E2%80% 9Cwhat-industry-wants-from-research%E2%80%9D/

[4] P. Kruchten, “A plea for lean software process models,” in

Proceedings of the 2011 International Conference on on Software and Systems Process, ser. ICSSP ’11. ACM, 2011, pp. 235–236.

[5] G. Lewis, E. Morris, and D. Smith, “Analyzing the reuse po- tential of migrating legacy components to a service-oriented architecture,” in Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Con- ference on, March 2006, pp. 9 pp.–23.

[6] H. M. Sneed, “Integrating legacy software into a service ori- ented architecture,” in Conference on Software Maintenance and Reengineering, 2006, pp. 3–14.

[7] J. Hutchinson, G. Kotonya, J. Walkerdine, P. Sawyer, G. Dob- son, and V. Onditi, “Evolving existing systems to service- oriented architectures: Perspective and challenges,” in Web Services, ICWS 2007. IEEE International Conference on, 2007, pp. 896–903.

[8] M. Razavian and P. Lago, “A Dashboard for SOA Migration,” 2011, under Submission.

[9] J. Meyer, “Service Oriented Architecture (SOA) Migra- tion Strategy for U.S. Operational Naval Meteorology and Oceanography (METOC),” in OCEANS 2007 - Europe, 2007.

[10] F. Cuadrado, B. Garcia, J. Dueas, and H. Parada, “A case study on software evolution towards service-oriented architec- ture,” in Advanced Information Networking and Applications, AINAW 08, 2008, pp. 1399–1404.

[11] R. Heckel, R. Correia, C. Matos, M. El-Ramly, G. Kout- soukos, and L. Andrade, “Architectural transformations: From legacy to three-tier and services,” Software Evolution, pp. 139–170, 2008.

[12] M. Razavian and P. Lago, “A Survey of SOA Migration in Industry,” in International Conference on Service Oriented Computing, ICSOC, 2011.

[13] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, “Legacy information systems: Issues and directions,” IEEE Software, vol. 16, pp. 103–111, 1999.

[14] E. J. Chikofsky and J. H. C. II, “Reverse engineering and design recovery: A taxonomy,” IEEE Software, vol. 7, no. 1, pp. 13–17, 1990.

[15] S. Tilley and D. Smith, “Perspectives on legacy system reengineering,” Reengineering Center Software Engineering Institute Carnegie Mellon University, Tech. Rep., 1995.

[16] M. Razavian and P. Lago, “Towards a Conceptual Framework for Legacy to SOA Migration,” in Fifth International Work- shop on Engineering Service-Oriented Applications (WE- SOA’09), 2010, pp. 445–455.

[17] R. Yin, Case Study Research, Design and Methods.Sage Publications, 1984.

[18] I. Seidman, Interviewing As Qualitative Research: A Guide forResearchersinEducationAndtheSocialSciences. Teach- ers College Press, 2006.

[19] M. B. Miles and M. Huberman, Qualitative Data Analysis: An Expanded Sourcebook(2nd Edition), 2nd ed. Sage Publications, Inc.

[20] I. Nonaka and H. Takeuchi, The knowledge-creating com- pany: How japanese companies create the dynamics of inno- vation. Oxford University Press, 1995.

[21] J. Creswell, Research design: qualitative, quantitative, and mixed method approaches. SAGE, 2003.

[22] E. R. Poort and H. van Vliet, “Architecting as a risk- and cost management discipline,” in Ninth Working IEEE/IFIP Conference on Software Architecture (WICSA), 2011.

[23] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, and N. Zazworka, “Managing tech- nical debt in software-reliant systems,” in Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, 2010, pp. 47–52.

Appendix J: Exloiting Codified User Task Knowledge to Discover Services

Konstantinos Zachos, Angela Kounkou, Neil Maiden

Abstract—Most methods and techniques for engineering service-based applications do not explicitly exploit knowledge about users and their tasks. However, codified knowledge about user tasks can be applied to improve service discovery and composition so that it is adapted to these tasks. This paper reports the application of user task models to improve requirements-based service discovery. It describes how user task models can be applied to enhance service discovery based on a catalogue of user tasks. More specifically it reports an evaluation of a new tool that demonstrate its potential utility when improving the discovery of web services for an e-government service-based application.

1 Introduction

Service-oriented computing increases the number of mechanisms through which software can adapt to its context [1]. Whilst established context factors such as time and location have been applied to the design of service-based applications [2], [3], one factor often overlooked in service-oriented computing is the user task. Service-based applications that invoke services adapted to the user task have the potential to enable the user to achieve the task goal more effectively than applications not adapted to the task. However, there has been little research to explore this potential.

User task modeling has been the subject of research in human-computer interaction since the 1980s. User task analysis [4] and models [5] are well-understood concepts. However, there have been few applications of user task models to the design or delivery of service-based applications, although exceptions do exist [6] in spite of the potential advantages that the use of such models can offer. Run-time service environments would be able to select, compose and invoke services that would explicitly fit with the goals and constraints of the user task, and design-time environments would be able to overcome ontological mismatches between user requests and descriptions of the software services to meet these requests [7]. For example, without knowledge of the user task such as *drive to a destination* and which classes of service to invoke to support a task, a motorist's need for an *accurate estimated time of arrival at a destination* cannot be associated to software service descriptions for *journey planning*, *weather forecasting* and *roadwork alerts*. Therefore, in the research reported in this paper, we investigated whether codified user task knowledge can deliver some of these potential advantages to the design and use of service-based applications.

Current approaches to designing service-based applications do not exploit codified knowledge about user tasks. Business process models and notations such as BPEL [8] and BPMN [9] indicate the process-oriented context in which services need to be invoked, however these models often lack important information about the actors performing the processes, and their goals, actions and constraints. Although initiatives such as BPEL4People [10] attempt to incorporate human considerations into the specification of business processes, they are limited to describing human activities as simple processes and do not codify knowledge about users and their tasks.

In contrast, codified user task models have the potential to provide different types of knowledge with which to improve the design of service-based applications. Examples of this knowledge include the end-state that the user is trying to achieve with the task, the different types of cognitive or interactive sub-task undertaken, and the concrete physical, financial and time resources needed to undertake each sub-task. Our research starts from the position that a design-time environment can exploit knowledge of these types to discover services meeting the user's goals and resource needs, compose services to support cognitive and interactive tasks more effectively, and invoke services that provide resources that users need.

In this paper we report the development and codification of user task models and their application to service discovery in one environment developed to design service-based applications. The user task models were developed at the class-level (e.g. *drive to a destination*) to maximize the leverage of each model during service discovery – one model could potentially be exploited during the design of all service-based applications that instantiate that task class (e.g. *drive from London to Paris via the*

Channel Tunnel). The codified user task models were documented in a searchable catalogue, then service queries were generated and fired at a service repository. An empirical evaluation explored the effect of modifying service queries with codified user task models on the precision and recall of one service discovery engine.

The remainder of this paper is in six sections. Sections 2 and 3 report current user task modeling and analysis approaches and their use in the development of service-based applications. Section 4 describes the new approach developed in the S-Cube project to exploit user task models in service discovery, then section 5 describes how user task models were codified and used in the design-time service discovery process. Section 6 reports the method and results from a multiphase evaluation study that investigated the effect of codified knowledge from user task models on the precision and recall of one service discovery engine. The paper ends with a review of the research hypotheses and threats to validity, and a discussion of the findings and future research.

2 User Task Models

A user task model is a description of the structured activities that are often executed by a user during the interaction with a system, influenced by its contextual environment, and performed to attain goals [11], [12]. Such models have been used extensively to support different phases of the software development life cycle, from requirement analysis to usability evaluation. During requirements work, for example, they can help analysts to understand how people perform their work [13] and describe how activities should be performed with a new application. Similarly, during design work, user task models can allow for software to be described more formally, analyzed in terms of usability, and be better communicated to people other than the analysts [14].

Different user task modeling techniques, semantics and syntax have been developed to support these different phases of the development life cycle [15], [16], [17], [18], [19], [20]. For example Hierarchical Task Analysis (HTA) [21] enables an analyst to model a task structure precisely during early design work. The more complex Task Knowledge Structures (TKS) approach [22], [23] also supports early design work through the description of two different parts of a task knowledge structure – the goal and taxonomic structures. The goal structure represents the sequencing of task activities, and the taxonomic structure models extensive knowledge about objects, their relationships and their behaviours.

A more recent approach is Concurrent Task Trees (CTT) [24], which provided additional semantics for describing sub-task types and precise temporal relationships between sub-tasks. One example CTT model describing the user task *Making a mobile phone call* is shown and described in [25]. The model enables the analyst to describe the overall task structure – for example that the *abstract* task *Handle communication* is divided into the *Connect to network* and *Use phone* sub-tasks. It describes different *user* sub-tasks, such as *Recall number*, *interaction* tasks such as *Switch on*, *Have conversation* and *Switch off*, and *application* tasks undertaken by the computerized device, for example *Connect to network* and *Show time-battery connectivity*. CTT also provides different semantics for expressing the temporal relations between sub-tasks, for example *enable with information passing* between the sub-tasks *enter PIN* and *connect to network*.

User task models developed with these and other similar approaches have been applied successfully in a wide range of domains for human-computer interaction, for example from applications in medical systems [26] to interactive TV listing guides [27]. The evidence points to the approach as an established means of describing and analyzing contexts related to tasks undertaken by users.

3 User Task Models for Service-Based Applications

Despite the potential role of user task models in the design and running of service-based applications, research on the topic is scarce and few published studies are available. Early work by Paterno et al. [28], [29] delivered an environment to support tasks and services matching with CTT task models to develop user interfaces. However the association between tasks and services was manually established, which limits the cost-effectiveness of the approach for the design of service-based applications. Later, Ruiz et al. [30] proposed a method with which to design web services that analysed user task descriptions to identify web application's required operations. Unlike Paterno's work, this approach was automated but did not include activities specific to the design of service-based applications – activities such as discovering, selecting and orchestrating software services. As such we argue that its role in service-based application design is more limited than it might otherwise be.

A more recent approach from Kritikos and Paterno [6] exploited a domain ontology and designer-provided user task models to produce service models specifying service combinations that realize application functionality. The approach generated service discovery queries from each system task description that were executed to discover, categorize and rank services based on a textual similarity measure. Although extended to activities specific to the design of service-based applications, and therefore more effective than the previously-reported approaches, we believe that the approach is not as cost-effective and widely applied as we would like. A designer is still required to design upfront user task models for each interaction with the envisaged system, which consumes resources and time unlikely to be available in increasingly short development cycles for service-based applications. Furthermore the service matching is dependent on a pre-existing domain ontology, thereby restricting the approach to applications for which a domain analysis has already taken place.

Whilst we conjecture that user task models have the potential to be used to improve the design of service-based applications, we need to ensure that applications exploiting these models are potentially cost-effective. For this to happen, we argue that the user task models should be amenable to the forms of automatic analyses common during service discovery, selection and composition, and their use possible without a priori domain analysis that restricts their scope of use. Indeed, greater leverage from a single user task model can be achieved if that model can be exploited to design applications in more than one domain.

4 User Task Models for Service Discovery

In this paper we report a new approach, supported by software tools, which exploits reusable user task models to enable context-aware service discovery and defines the context as the modeled user task. The approach was developed as part of S-Cube, the EU-funded Network of Excellence for Software Services [31]. It reuses knowledge from application domain-independent user task models, similar to the approach in KADS [31], to provide user task knowledge missing from other published service discovery approaches e.g. [32], [33], [34]. It seeks to overcome the semantic mismatch between problem queries and solution service descriptions.

The approach is depicted graphically in Figure 4_1. The task-based service discovery algorithm reuses knowledge about classes of service solution linked to tasks modeled in user task models to reformulate a service query – terms describing the user's goals, tasks and resources, are extended or replaced with terms that describe classes of software services that might be relevant, thereby increasing the likelihood of discovering them. For example, a user task model describing the journey undertaken by a driver to reach a destination would replace terms in service queries such as *plan*, *travel*, *route* and *help* (terms in the application-independent domain of the user task) with terms such as *journey planner*, *routing*, *parking space*, and *global positioning system*, which are terms describing relevant classes of software service for the user task.

Reusing user task knowledge from models to reformulate service queries can, we hypothesise, improve the levels of precision and recall of service discovery over the levels achieved with existing keyword matching and information retrieval techniques. User task knowledge can be utilized both to add new terms to service queries more likely to result in retrieval of relevant services and to filter out terms likely to result in the retrieval of irrelevant services. Although applicable to support service discovery at run-time, we first investigated the effect of the approach on the precision and recall of service queries generated early in the design process – queries that encapsulate user requirements – to investigate four hypotheses.

We posited one concrete hypothesis to explore the relative effectiveness of different strategies with which to add new user task knowledge to service queries. Put simply, new user task knowledge can be used either to add new terms to existing terms in a service query or to replace existing terms. We hypothesized that adding service query terms would be more effective at design-time because early design is a divergent activity, and more query terms that have the same or similar meanings to terms describing software services will lead to retrieval of more relevant services.

Another three concrete hypotheses were posited to explore the relative effectiveness of the new approach over an original service discovery approach upon which it was based. Again, put simply, the hypotheses investigated the effect of the introduction of codified user task knowledge on the precision and recall of selected service queries. We hypothesized that reformulating service queries with terms describing sought service classes related to classes of task would increase relevant and decrease irrelevant services that were retrieved because more relevant and fewer irrelevant terms were included in service queries.

More precisely the four hypotheses, labelled **H1** to **H4**, were:

H1: Extending rather than replacing service queries with additional knowledge about user tasks will *improve the overall effectiveness of service discovery;*

H2: The reformulation of service queries with knowledge about user tasks will *decrease the number of irrelevant services* retrieved by a service discovery engine;

H3: The reformulation of service queries with knowledge about user tasks will *increase the number of relevant services* retrieved by a service discovery engine;

H4: The reformulation of service queries with knowledge about user tasks will *improve the overall correctness of services* retrieved by a service discovery engine.

Each hypothesis was investigated empirically in experiments in which the new approach was compared directly to an original service discovery approach using a predefined set of service queries. The original approach implemented both common keyword matching and a more sophisticated information retrieval technique.

5 Codified User Task Models for Service Discovery

This section describes in detail the new approach and how it was developed on top of one existing service discovery engine. It summarizes the original approach, then describes how we generated and specified user task models and extended the original service discovery engine with new knowledge from user task models in the new approach.

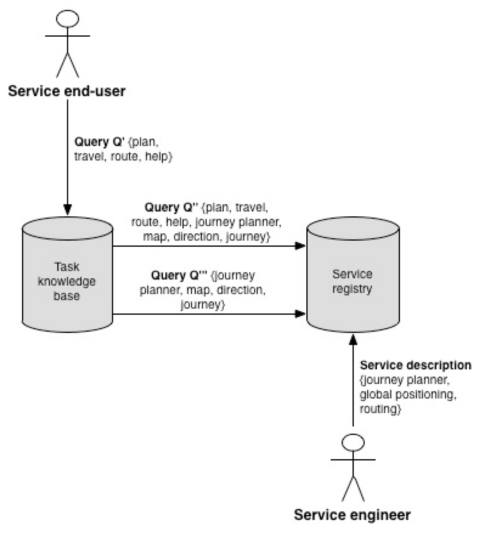

Service end-user
Query Q' {plan, travel, route, help}
Query Q'' {plan, travel, route, help, journey planner, map, direction, journey}
Query Q''' {journey planner, map, direction, journey}
Task knowledge base
Service registry
Service description {journey planner, global positioning, routing}
Service engineer

Fig. 4_1. Task-based service discovery process

5.1 The Original Service Discovery Approach

We selected the SeCSE service discovery environment [35] as the original approach upon which to design and implement the S-Cube approach, and therefore compare it against. The original service discovery approach developed to support design-time acitivities was called the *Expansion and Disambiguation Discovery Engine*, or EDDiE for short [35], and had been subjected to evaluations that demonstrated its effectiveness on the design of a range of service-based applications [36].

EDDiE was originally developed to overcome several service discovery challenges. Previous research prior to EDDiE – for example *SWWS*, *DIP*, *ASG*, *InfraWebs* and *NeP4B* – demonstrated that semantic descriptions such as *SAWSDL* [37], *MicroWSMO* [38], *WSMO* [39] and *OWL-S* [40] could be used to enable more precise discovery of services. However, although these approaches focused on automation support and flexible matching between descriptions of services and queries, they did not address the problem of the heterogeneity of the vocabularies used in service descriptions and user requirements. Therefore EDDiE deployed established information retrieval techniques to formulate service queries from use case and requirements specifications expressed in structured natural language [41]. It can be configured to use either these information retrieval techniques or simpler keyword matching techniques [42], [43]. The former we call Full-EDDiE, the latter EDDiE-lite. Each is described in turn. Both were used as baselines with which to exploit codified user task knowledge in service discovery.

5.1.1. The Full-EDDiE Algorithm

Full-EDDiE adds domain knowledge to service queries to overcome the problem of vocabulary heterogeneity with service descriptions. It has two important capabilities derived from established information retrieval techniques:

1. *Query expansion* – the addition of terms in the query that have the same or similar meaning to existing query terms, to make the query more complete;
2. *Term disambiguation* – selecting the meaning, or sense of each term in the query to enable query expansion, thus making the query unambiguous.

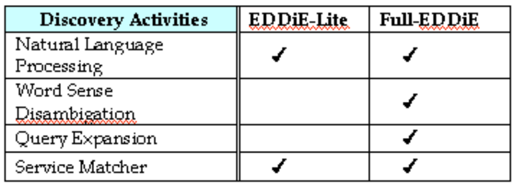
These capabilities were implemented with four components; the *Natural Language Processing*, *Word Sense Disambiguation*, *Query Expansion* and the *Service Matching*. In the first the service query was divided into sentences, then tokenized and part-of-speech tagged and modified to include each term's morphological root (e.g. *driving* to *drive*, and *drivers* to *driver*). Secondly, the algorithm applied procedures to disambiguate each term by defining its correct sense and tagging it with that sense (e.g. defining a *driver* to be a *vehicle* rather than a *type of golf club*). Thirdly, the algorithm expanded each term with other terms that have similar meaning according to the tagged sense, to increase the likelihood of a match with a service description (e.g. the term *driver* is synonymous with the term *motorist* which is also then included in the query). In the fourth component the algorithm matched all expanded and sense-tagged query terms to a similar set of terms that describe each candidate service, expressed using a *service description* facet [44]. Query matching is in 2 steps: (i) XQuery text-searching functions to discover an initial set of services descriptions that satisfy global search constraints; (ii) traditional vector-space model information retrieval, enhanced with WordNet, to further refine and assess the quality of the candidate service set. This two-step approach overcame XQuery's limited text-based search capabilities.

The WordNet on-line lexicon fulfilled an important role for three of the algorithm's components. WordNet is a lexical database inspired by current psycholinguistic theories of human lexical memory [45]. It has two important features. The first divides the lexicon into four categories: nouns, verbs, adjectives and adverbs. Word meanings, called senses, for each category are organized into synonym sets (synsets) that represent concepts, and each synset is followed by its definition or gloss that contains a defining phrase, an optional comment and one or more examples. In the second WordNet is structured using semantic relations between word meanings that link concepts. Relationships between concepts such as hypernym and hyponym relations are represented as semantic pointers between related concepts [45]. A hypernym is a generic term used to designate a whole class of specific instances. For example, *vehicle* denotes all the things that are separately denoted by the words *train*, *chariot*, *dogsled*, *airplane*, and *automobile*, and is therefore a hypernym of each of those words. On the other hand, a hyponym is a specific term used to designate a member of a class, e.g. *chauffeur*, *taxidriver* and *motorist* are all hyponyms of *driver*. A semantic relation between word meanings, such as a hypernymy, links concepts.

5.1.2. The EDDiE-Lite Algorithm

The *EDDiE-Lite* algorithm only implements two of the four components – *Natural Language Processing* and *Service Matching*. As such it delivers the functionality of current keyword-based service discovery approaches (e.g. [42], [43], [46]), i.e. taking as inputs queries made up of keywords, and matching the keywords from the query against those used to describe services they access in a catalogue or registry. EDDiE-Lite provides a second, lower baseline against which to judge the effectiveness of codified user task knowledge in service discovery.

TABLE 5_0 DISCOVERY ACTIVITIES ASSOCIATED WITH FULL-EDDIE AND EDDIE-LITE



Discovery Activities	EDDiE-Lite	Full-EDDiE
Natural Language Processing	✓	✓
Word Sense Disambigation		✓
Query Expansion		✓
Service Matcher	✓	✓

A comparison between the Full-EDDiE and EDDiE-Lite discovery activities is listed in Table 5_0. Full-EDDiE undertakes service discovery with *term expansion* and EDDiE-Lite undertakes it with *no term expansion*.

5.2 The TEDDiE Algorithm and User Task Models

Results from previous evaluations of EDDiE [36] indicated that query expansion and term disambiguation alone, even in the sophisticated form reported, could not overcome all of the semantic mismatches between user requests and software service descriptions. We hypothesized that user task context knowledge was needed. Therefore to provide this knowledge we extended both versions of EDDiE with a catalogue of class-level user task models that link *application* sub-tasks to classes of service solution. A new algorithm was implemented to discover services by first matching terms describing the user problem to user task models that, in turn, are used to reformulate the service queries with terms describing classes of services linked to relevant *application* sub-tasks in the user task model. These reformulated service queries are then fired at a service registry.

This new approach – a task-based extension to the EDDiE algorithm – was called TEDDiE. As with EDDiE, two versions were developed, one to use information retrieval techniques that adds domain knowledge to service queries, the other simple keyword-matching. The former we called Full-TEDDiE, the latter TEDDiE-lite. Full-TEDDiE was implemented with information retrieval techniques so that it could be compared directly to Full-EDDiE, whilst TEDDiE-Lite list was implemented with keyword matching so that it could be compared directly with EDDiE-Lite.

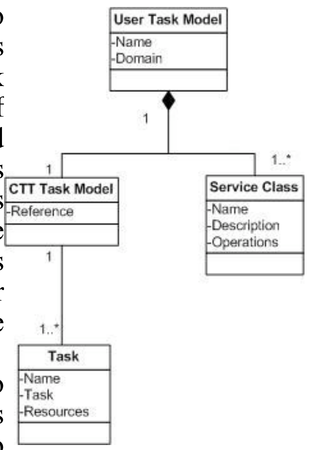


Fig. 5_1. Outline task model schema expressed as a UML class diagram

Both first versions of TEDDiE were implemented with class-level user task models in the *e-government* domain. Each could be applied to describe user interactions with a wide range of service-based applications ranging from *self health diagnosis* and *planning a shopping trip* to *vaccinating children* and *going on holiday*. The experiments reported in Section 6 use service queries drawn from one of these applications in which an end-user uses a government journey planning service to *plan journeys* and *obtain weather updates*.

The next two sections describe how we codified user task models as CTT models and implemented a new service discovery approach with these models.

TABLE 5_1
CALCULATE DISTANCE TASK

Name	Calculate distance
Task	Compute and output the distance between two specified geographical locations "A" and "B"
Goal	Coordinates for the geographical locations

5.2.1 Codifying User Task Models as CTTs

We chose to represent the user task models in the catalogue using the CTT task modeling formalism [12] because CTT adapts an engineering approach to user task models. The more complete and precise semantics of CTT has the potential to support automated service discovery more effectively than other user task modeling formalisms.

The codification of the user task models was performed manually in a two staged process: In the first each user task model was specified by completing a set of templates that guided the description of user task elements in text form and user task models in graphical form, and to map subtasks to classes of software services. In the second each specified user task model was uploaded into the online catalogue.

5.2.2 The Structure of a User Task Model

A user task model defines a reusable task structure that encapsulates well-defined functionality for a recurrent design problem [21]. Figure 5_1 specifies the schema of the user task models in the catalogue. Key elements of the schema are described.

User task and goal descriptions: each user task was specified with natural language descriptions of the task in context and the user goal achieved by completing the task. For example the task and associated user goal for the user task *Calculate a distance* are described in natural language as shown in Table 5_1.

The CTT model: the task structure was expressed using the CTT formalism [24]. Each sub-task specified in the CTT model could be an: (i) *abstract* task that is decomposed further; (ii) *user* task undertaken by the user; (iii) *interaction* task carried out by the interaction of a user with a software system, or: (iv) *application* task fully undertaken by software [18]. Possible sequences of these sub-task types were described by CTT operators such as *concurrent* and *enable with information passing* [13]. The *Calculate a distance* task is specified graphically as a CTT task model in Figure A_1 of Appendix A.

The task hierarchy is described in a tree structure using task decomposition. The higher level task *Calculate distance* is decomposed into lower level subtasks that execute it: *Input start*, *Enter destination*, *Submit data*, *Validate data*, *Compute distance*, *Display distance* and *View distance*. Two of these subtasks are themselves decomposed further – *Input start* comprises *Detect current location*, *Accept current location* and *Enter start*; and *Validate data* is decomposed into *Check match with current locations*, *Request re-input* and *Confirm data validity*. Graphical syntax elements indicate each of the tree nodes' types: each task involved in this example falls into one of three categories. Application tasks ( ) comprise *Validate data*, *Compute distance*, *Display distance*, *Detect current location*, *Check match with current locations*, *Request re-input* and *Confirm data validity*, all of which are executed by a software element without explicit intervention from users. Interaction tasks ( ) on the other hand comprise *Enter destination*, *Submit data*, *View distance*, *Accept current location* and *Enter start*, all activities that require interaction between a user and a system in order to occur, for instance to obtain the necessary task resources. The remaining tasks *Calculate distance* and *Input start* are represented as abstract tasks ( ) since they require complex actions and cannot neatly fall into either of the other task categories of CTT.

Relationships between tasks at a same hierarchical level and their occurrence in time are described using temporal operators. For example *Input start* and *Enter destination* are independent tasks ([]) as they do not require information flow from each other, and can theoretically occur in any order. However, *Enter destination* enables and passes on information (the *destination elicited*) to *Submit data*, which in turn enables and inform *Validate data*. This same relationship - "enable with information passing" ([]>>) – also occurs between the pairs *Validate data* and *Compute distance*, and *Compute distance* and *Display distance*: this in effect passes on the data resulting from a task's processes to the next sequential task for further use. The last temporal operator at this hierarchical level (|||) designates *Display distance* and *View distance* as concurrent tasks. At the lowest hierarchical level, the *Input start* sub-tree represents *Detect current location* as enabling with information passing *Accept current location* and the operator ([]) indicates this as an option to the task *Enter start* occurring. The other sub-tree (stemming from *Validate data* this time) represents *Check match with current locations* enabling (>>) *Request re-input* as an option ([]) to *Confirm data validity* occurring.

Associations to service classes: each user task model associated each *application* subtask described in a CTT model to one or more classes of software service. One way of associating user task models to service classes was to investigate the original design rationale for implemented software services, however such rationale are rarely available and complete. Another way was to associate service classes to user task models based on systematic analysis by people with domain expertise. This was the way adopted in the new approach. A small team of researchers with knowledge of the user task models, the CTT formalism and the domains in which users would normally undertake these classes of task, systematically explored each pair-wise association between a user task and service class and made a design decision as to whether an *application* sub-task could be wholly or partially implemented using a software service of that class. An association was created between the sub-task and service class if enhancement was agreed to take place.

The result was that each application sub-task could be associated with zero, one or many service classes, and each service class could be associated with one or many application sub-tasks. Returning to our example with the *application* sub-task *Validate data* depicted in Figure A_1 of Appendix A, we associated it to services of the class *DataValidation* because one or more such services could be reasonably invoked by a service-based application.

Description of each service class: we sought to describe each service class associated with a user task model using neutral terms to avoid unintended bias during service discovery. Each service class's name, function and operations were described using terms sourced from online encyclopedia, namely Wikipedia [47], Encyclopedia Britannica [48] and the more specialized Webopedia computer dictionary [49]. Continuing the *DataValidation* example from Figure A_1 of Appendix A, Table 5_2 reports the *DataValidation* class' functional description, taken from the source concept definition, and list of operations derived from the action terms or verbs contained in the concept description.

The next section describes how these codified user task models were used by both versions of TEDDiE to add this user task knowledge to service queries.

5.2.2 Task-Based Service Discovery wth TEDDiE

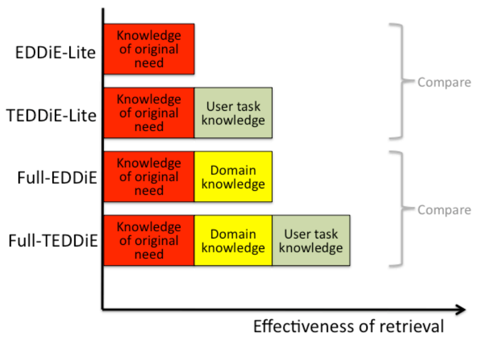
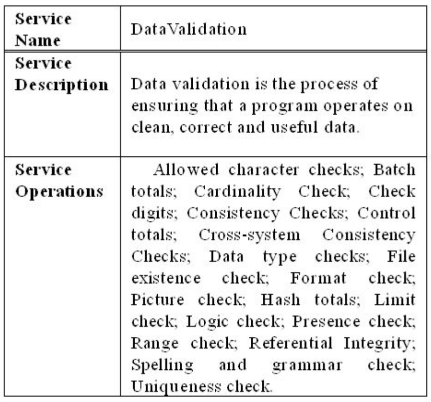


Fig. 5_3. Sources of knowledge forming the queries generated by EDDiE andTEDDiE

To develop TEDDiE we extended the original EDDiE algorithm with two new steps that accessed a catalogue of user task models and used model content to reformulate service queries:

1. *Match to user task models*: EDDiE matched a service query to the natural language description of each user task model in the catalogue. Full-EDDiE implemented it with *term expansion* and EDDiE-Lite implemented it with *no term expansion*. The result was an ordered set of retrieved user task models;
2. *Reformulate service query*: TEDDiE extracted terms from the descriptions of service classes associated with each *application* sub-task in each retrieved user task model to generate new service queries. Reformulation implemented two activities: (i) *extend* the original service query with the new terms extracted from the user task model; (ii) r*eplace* the service query with the new terms extracted from the user task model.
3. *Service match*: EDDiE used each reformulated service query to discover candidate service specifications in service registries. Again this was implemented in two ways. Full-EDDiE implemented it with *term expansion* while EDDiE-Lite implemented it with *no term expansion*. The result was an ordered set of service specifications that match to the revised service query.

TABLE 5_2
THE DATA VALIDATION SERVICE CLASS



Service Name	DataValidation
Service Description	Data validation is the process of ensuring that a program operates on clean, correct and useful data.
Service Operations	Allowed character checks; Batch totals; Cardinality Check; Check digits; Consistency Checks; Control totals; Cross-system Consistency Checks; Data type checks; File existence check; Format check; Picture check; Hash totals; Limit check; Logic check; Presence check; Range check; Referential Integrity; Spelling and grammar check; Uniqueness check.

To investigate our research hypotheses we implemented the four specific versions of TEDDiE alongside the two original versions of EDDiE. The sources of knowledge forming the queries generated by these six versions is shown in Figure 5_3.

EDDiE-Lite generated service queries containing only terms extracted directly from the original statement of need. TEDDiE-Lite either replaced or extended these terms with terms extracted from descriptions of service classes in matched user task models. Full-EDDiE added domain knowledge terms in on-line thesauri to terms extracted from the original statements. Full-TEDDiE added terms extracted from both domain knowledge in on-line thesauri and descriptions of service classes in matched user task models to either extend or replace the terms extracted from the original statements of need.

We chose not to mix information retrieval and keyword matching activities in either new version of TEDDiE because we wanted to investigate our four research hypotheses – the effect of user task knowledge on the outcomes of service discovery.

Table 5_3 specifies the six concrete combinations of activities – called *strategies* – implemented in the six versions of EDDiE and TEDDiE. The increasingly sophisticated service discovery strategies are labeled A-F to ease of reference during the results section.

For example, consider the following initial service query:

The user sends a journey planning request with details about the start, end point and travel preferences for his journey.

In the first step of the strategy implemented by Full-TEDDiE the algorithm extracts query terms from the original service query, i.e.

Q' = [journey, travel, preference, user, start, end point, plan, send],

as well as generate new query terms after the application of term expansion as described in Section 5.1, for example:

Q'' = [termination, commence, direction, move, place, go].

The query consisting of both *Q'* and *Q''* is then matched to the catalogue of user task models. Assume that one of the retrieved models is C*alculate a distance.* One of the classes of service associated with one of its *application* sub-tasks is *Route planning software.* The class has the following functional description:

Route planning software is a computer software programme, designed to plan a (optimal) route between two geographical locations using a journey planning engine, typically specialised for road networks as a road route planner. It can typically provide a list of places one will pass by, with crossroads and directions that must be followed, road numbers, distances, etc. It also usually provides an interactive map with a suggested route marked on it.

With information about this service class Full-TEDDiE implements *extend* query reformulation that then generates the following original and expanded service class terms:

S' = [direction, crossroads, location, distance, road, map, suggest]

S'' = [way, itinerary, calculation, travel by, path, motor, travel]

In the final step TEDDiE then generates and fires a reformulated service query based on the *extend* query reformulation, i.e.

Q = {Q', Q'', S', S''}

or more concretely

Q = {[journey, travel, preference, user, start, end point, plan, send], [termination, commence, direction, move, place, go], [direction, crossroads, location, distance, road, map, suggest], [way, itinerary, calculation, travel by, path, motor, travel]}.

6 First Evaluations of TEDDiE

TABLE 5_3
DISCOVERY ACTIVITIES ASSOCIATED WITH EACH DISCOVERY STRATEGY

		Strategies					
		A	B	C	D	E	F
TEDDiE step	**Discovery activities**	No Term Expansion			Term Expansion		
		EDDiE-Lite	TEDDiE-Lite		Full-EDDiE	Full-TEDDiE	
			Replace	Extend		Replace	Extend
1	Natural language processing of original service query	▯	▯	▯	▯	▯	▯
1	Word sense disambiguation of original service query				▯	▯	▯
1	Query expansion of original service query				▯	▯	▯
1	Match query to user task models		▯	▯		▯	▯
2	Natural language processing of each service class description		▯	▯		▯	▯
2	Word sense disambiguation of terms of each service class description					▯	▯
2	Query expansion of terms of each service class description					▯	▯
2	Reformulating service queries by replacing terms		▯			▯	
2	Reformulating service queries by extending terms			▯			▯
3	Service matching with reformulated service queries	▯	▯	▯	▯	▯	▯

We undertook two evaluations of TEDDiE during a series of design-time service discovery activities to investigate the four hypotheses H1, H2, H3 and H4 reported in Section 4. In the first evaluation, to investigate hypothesis H1 and determine TEDDiEs' more effective strategies, we implemented the four strategies B, C, E and F in Table 5_3 to investigate the effect on *relevant* and *irrelevant* service(s) retrieved from a service registry. In the second, to investigate hypotheses H2, H3 and H4 and the effect of adding user task knowledge to service queries, we investigated the effect on *relevant* and *irrelevant* service(s) retrieved from a service registry by EDDiE and TEDDiE using the strategies A, C, D and F. Strategies B and E had been excluded based on the results from the first evaluation.

6.1. Evaluations Method

The two evaluations were undertaken in four stages. In the first stage a set of user task models extended with knowledge about classes of software service as described in Section 5.2 was developed. In the second stage human expert judgement was used to classify the services in a target service registry that a series of pre-defined service queries should retrieve as *relevant*. In the third each service query was fired at the target service registry with the corresponding DS. In the final stage we applied statistical analyses to the collect results to investigate each hypothesis. Each stage, and the target service registry, is described in turn.

6.1.1 The Target Service Registry

The target service registry for both evaluations was a version of a SeCSE service registry containing descriptions of 215 existing web services in domains ranging from flight booking and weather reporting to route planning [44]. The SeCSE registry was developed in 2006 to describe different aspects of a service stored in it using ten facets including the *service signature*, *description* and *quality-of-service* [44]. Each facet is described using an XML data structure linked to the UDDI service entry and accessed using Java APIs. A set of SeCSE APIs with WSDL interfaces allow access to the UDDI registries and service facets. The registry itself is implemented using eXist, an Open Source native XML database featuring index-based XQuery processing, automatic indexing and tight integration with existing XML development tools. The EDDiE and TEDDiE service discovery algorithms access and match information stored in the *description* facet composed of attributes with structured natural language values describing, for example, *service goal* and *service behaviour*. A full description of the XML structure of this facet is provided in [44].

6.1.2 Extending User Task Models with Classes of Software Services

We populated the user task model catalogue with codified user task models relevant to the evaluation. Seven were codified for tasks that most citizens would be expected to undertake from time to time:

- *Obtain health advice*
- *Access a patient record*
- *Request a route*
- *Calculate a distance*
- *Park a car*
- *Zoom into a map*
- *Obtaining a weather forecast*

Each user task model was codified so that it was consistent with the structure reported in Section 5.2.2:

A. A natural language description of the overall user task and associated user goal;
B. A full and codified CTT model;
C. Associations between each of these service classes and one or more *application* sub-tasks specified in the CTT model;
D. Natural language descriptions of one or more of these service classes.

The full specification of the seven user task models is included in Appendix A. In total, 33 service classes were associated with the 7 user task models. Table 6_1 shows the association sub-tasks of the *request route* user task model to service classes.

6.1.3 Classification of Candidate Services

TABLE 6_1
ASSOCIATIONS MADE BETWEEN SUB-TASKS OF THE REQUEST ROUTE USER TASK MODEL TO SERVICE CLASSES

Service Classes / Request route sub-tasks	Choice	Data type checks	Data validation	Display device	Document retrieval	Global positioning system	GPS tracking unit	Input	Input device	Journey planner	Output	Perception	Query	Real time locating systems	Route planning software	Routing	Tracking system	Validation methods	Vehicle tracking system
Input start								□	□										
Detect current location						□	□							□			□		□
Display current location				□							□								
Accept current location																			
Enter start location								□	□										
Enter destination								□	□										
Select route preferences																□			
Submit query																			
Validate query		□	□															□	
Confirm validity																			
Request data re-input																			
Compute possible routes										□					□				
Display route suggestions				□	□						□								
Select route	□																		
Select from suggestions	□																		
Launch new query													□						
Display route											□								
View route												□					□		

Human expert judgement was used to select the services in the target service registry that the pre-defined service queries should retrieve as *relevant*. The service queries were, in turn, derived from use case specifications specified according to the SeCSE service discovery process [41]. The first use case specified the required behaviour when a citizen undertakes journey planning, and the second the behaviour when requesting a weather forecast. The concrete input service queries are described in Appendix B.

We identified the services in the target registry to be retrieved as relevant by each service query from 12 human judges, all of whom were experts in software services applied to e-government. Each assessed the relevancy of each service description for each use case specification. The human judges determined that 37 of the 215 available services should be discovered for the journey planning use case, and 33 for the weather forecasting use case. These services are listed in Appendix C. These judgments provided the *classified relevant services* with which to assess EDDiE and TEDDiE.

6.1.4 Evaluation Measures

We fired a total of 92 service queries at the target service registry. The number of generated EDDiE service queries was lower than for TEDDiE because of the presence of multiple sub-tasks in user task models used to generate multiple service queries.

Each *discovered service* was compared manually against the list of *classified relevant services* for the use case specification from which each service query was generated. If the service was on the list, then the service was categorized as a *retrieved relevant service*. The totals of *classified relevant services*, *discovered services* and *retrieved relevant services* where then analyzed using established measures of precision and recall from information retrieval research [50], [51]. Using these measures the precision for each service query was defined as:

*(Total retrieved relevant services / Total discovered services)*100*

The recall for each service query was defined as:

*(Total retrieved relevant services / Total classified relevant services)*100*

Finally the balanced F-score for each service query was defined as:

*(2*Precision*Recall / Precision + Recall)*100*

One precision measure, one recall measure and one balanced F-score measure was generated for each of the 92 service queries fired at the service registry, providing a total of 276 precision, recall and balanced F-score measures. Of the 92 service queries 88 were generated by TEDDiE and 4 by EDDiE, and 46 were generated by Full-EDDiE and Full-EDDiE and 46 TEDDiE and TEDDiE-Lite. Results are reported in the next section.

TABLE 6_3
PAIRED T-TESTS ON TEDDIE DISCOVERY STRATEGIES

Paired t-test statistics	TEDDiE-Lite with *extend* versus *replace*		Full-TEDDiE with *extend* versus *replace*	
	P-value	Difference statistically significant?	P-value	Difference statistically significant?
Precision	0.4182	no	0.3312	no
Recall	0.0002	yes	0.0012	yes
F-score	0.0011	yes	0.0115	yes

6.2 Evaluation Results

We analyzed the totals of retrieved *relevant* and *irrelevant* services for each of the 92 service queries and 276 computed precision, recall and balanced F-score measures to investigate each hypothesis in turn. First we analyzed the totals and measures to explore the relative effectiveness of the four TEDDiE strategies B, C, E and F in Table 5_3.

6.2.1 Evaluating TEDDiE's Most Effective Strategies

Summaries of the results generated by the service queries generated using each of the four strategies are shown in Table 6_2. Full-TEDDiE with *extend* resulted in the largest number of retrieved *relevant* services but also the largest number of retrieved *irrelevant* services. An arithmetic mean was used to calculate average totals of services retrieved by the queries generated with each strategy. The balanced F-score measures indicated that the two *extend* strategies were more effective than the two *replace* strategies (51% and 53% to 31% and 37%) but the effect of *term expansion* against *no term expansion* was minimal (51% and 31% to 53% and 37%). The arithmetic means of the precision measures for the four strategies varied little but the means of the recall measures revealed that *extend* led to better recall than *replace* (72% and 59% to 41% and 28%): the strategy with *term expansion* and *extend* retrieved the highest average mean of services per query.

Paired t-test statistical analyses were undertaken as shown in Table 6_3 to analyze the effect of *extend* versus *replace* and *term expansion* versus *no term expansion.*

Although precision for TEDDiE-Lite and Full-TEDDiE with *extend* was not a significant improvement over precision for TEDDiE-Lite and Full-TEDDiE with *replace* (t=0.87, p=0.4182 and t=1.23, p=0.3312 respectively), the balanced F-score measures revealed that queries generated with *extend* had significantly higher overall effectiveness than queries generated with *replace* (t=5.40, p=0.0011 and t=5.99, p=0.0115 respectively). One reason appears to be better recall by queries generated with *extend*: TEDDiE-Lite with *extend* achieved significantly higher recall measures (t=8.77, p=0.0002) whilst the difference in the recall measures between Full-TEDDiE with *extend* and Full- TEDDiE with *replace* achieved similar results (t=7.22, p=0.0012).

In conclusion this first evaluation revealed that TEDDiE *extending* service queries was more effective than TEDDiE *replacing* terms in service queries. Therefore we explored how the two versions of TEDDiE with *extend* performed against the two equivalent versions of EDDiE – the strategies A, C, D and F in Table 5_3.

Comparing TEDDiE and EDDiE

We investigated the effectiveness of TEDDiE against EDDiE using the computed arithmetic means of the totals of *relevant* and *irrelevant* retrieved services and the precision, recall and balanced F-score measures of service queries generated with the four strategies. These measures were used to investigate whether TEDDiE and TEDDiE-Lite decreased the number of *irrelevant* services, increased the number of *relevant* services, and improved the overall effectiveness against EDDiE and EDDiE-Lite respectively.

The arithmetic means of the totals of *relevant* and *non-relevant* retrieved services for all service queries are in Table 6_4.

TABLE 6_2
OVERALL PERFORMANCE OF EACH DS THAT IMPLEMENT TEDDIE ALGORITHM

Statistical Dimensions	TEDDiE-Lite		Full-TEDDiE	
	No Term Expansion		Term Expansion	
	Replace	Extend	Replace	Extend
Number of retrieved services	25	47	39	61
Number of *relevant* services	10	21	15	25
Number of *non-relevant* services	15	26	25	36
Precision	43%	46%	40%	42%
Recall	28%	59%	41%	72%
F-score	31%	51%	37%	53%

Results revealed that Full-TEDDiE and TEDDiE-Lite retrieved both more *relevant* services but also more *irrelevant* services than Full-EDDiE and EDDiE-Lite respectively. The balanced F-score measures of overall effectiveness revealed that service queries with *no term expansion* were higher (51% to 41%) whilst there was much less difference for service queries with *term expansion* (53% to 54%). We then performed paired t-tests to compute p-values to either reject or accept each *null hypothesis* using the precision, recall and balanced F-score measures for each of the 92 service queries. Results are shown in Table 6_5.

The balanced F-score measures revealed that TEDDiE-Lite had significantly higher overall effectiveness than EDDiE-Lite (t=4.08, p=0.0362) but Full-TEDDiE did not have significantly higher overall effectiveness than Full-EDDiE (t=0.69, p=0.7575). One reason appears to be better recall by TEDDiE: TEDDiE-Lite achieved significantly higher recall measures (t=6.44, p=0.0045) whilst the difference in the recall measures between Full-TEDDiE and Full-EDDiE neared significance (t=3.76, p=0.0538). However TEDDiE-Lite's precision was not a significant improvement over EDDiE-Lite (t=1.13, p=0.5175). Indeed, Full-EDDiE achieved significantly higher precision scores than Full-TEDDiE (t=5.75, p=0.0166). The results are depicted graphically in Figure 6_1. Because extending service queries with terms from user task models had significantly increased the recall measures we investigated whether the number of terms in a query had any effect on recall. We used *Pearson's correlation coefficient (r)* to measure the strength of association between *query length* and *recall measure*. Results in Table 6_6 revealed that only 56% (r^2) of the variation in the number of query terms was related to the variation in the recall values, and the correlation coefficient was not significantly different from zero (co-efficient score, p<0.1446). In short there was no evidence that the number of terms in a query increased recall measures.

TABLE 6_5
P-VALUES FROM PAIRED T-TESTS

Paired t-test statistics	No Term Expansion		Term Expansion	
	P-value	Difference statistically significant?	P-value	Difference statistically significant?
Precision	0.5175	no	0.0166	yes
Recall	0.0045	yes	0.0538	yes
F-score	0.0362	yes	0.7575	no

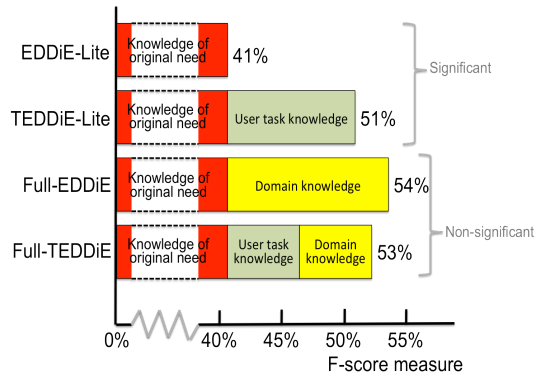


Fig. 6_2. Key results depicted graphically when comparing TEDDIE and EDDiE

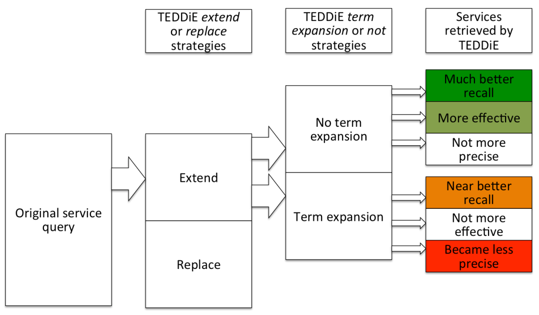


Fig. 6_1. Results depicted graphically

In conclusion our results revealed the importance of the expansion of queries on TEDDiE's performance. With *no term expansion* TEDDiE-Lite generated higher recall and balanced F-score measures than EDDiE-Lite, but with *term expansion* Full-EDDiE generated higher precision measures than Full-TEDDiE. The key results are depicted graphically in Figure 6_2.

TABLE 6_4
OVERALL PERFORMANCE OF EDDIE AND T-EDDIE STRATEGIES

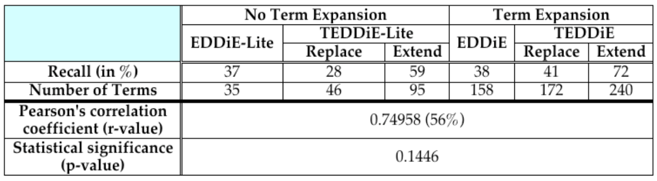
	No Term Expansion		Term Expansion	
	EDDiE-Lite	TEDDiE-Lite	Full-EDDiE	Full-TEDDiE
Number of retrieved services	27	46	36	62
Number of *relevant* services	12	21	19	25
Number of *non-relevant* services	15	25	17	37
Precision	50%	46%	55%	42%
Recall	37%	59%	38%	72%
F-score	41%	51%	54%	53%

In other words adding only user task knowledge to service queries generated higher recall and balanced F-score measures than not adding user task knowledge, but adding both domain and user task knowledge to service queries generated lower precisions scores than adding only domain knowledge.

6.3 The Hypotheses Revisited

The reported results were used to accept or reject each of

TABLE 6_6
NUMBER OF QUERY TERMS COMPARED TO RECALL FOR EACH STRATEGY



	No Term Expansion			Term Expansion		
	EDDiE-Lite	TEDDiE-Lite		EDDiE	TEDDiE	
		Replace	Extend		Replace	Extend
Recall (in %)	37	28	59	38	41	72
Number of Terms	35	46	95	158	172	240
Pearson's correlation coefficient (r-value)	0.74958 (56%)					
Statistical significance (p-value)	0.1446					

the four hypotheses under investigation:

H1 That extending rather than replacing service queries with additional knowledge about user tasks will *improve the overall effectiveness of service discovery* was **accepted** because the balanced F-score measures for *extend* strategies were significantly higher than for *replace* strategies;

H2 That the reformulation of service queries with knowledge about user tasks will *decrease the number of irrelevant services* retrieved by a service discovery engine was **rejected** because the mean average precision measures generated with TEDDiE were lower than the equivalent measures with EDDiE;

H3 That the reformulation of service queries with knowledge about user tasks will *increase the number of relevant services* retrieved by a service discovery engine was **partially accepted** because the recall measures generated with TEDDiE-Lite were significantly higher than the equivalent measures with EDDiE-Lite. However this was not the case between Full-TEDDiE and Full-EDDiE;

H4 That the reformulation of service queries with knowledge about user tasks will *improve the overall correctness of services* retrieved by a service discovery engine TEDDiE will improve the overall correctness of services retrieved was **partially accepted** because the balanced F-score measures for TEDDiE-Lite were significantly higher than for EDDiE-Lite. Again however this was not the case between Full-TEDDiE and Full-EDDiE.

In short, TEDDiE was more effective than EDDiE only when it *extended* simple service queries originally formed of original terms. Why was this?

The acceptance of hypothesis **H1** demonstrates that adding user task knowledge to service queries was more effective, and suggests that these original terms were still needed to retrieve *relevant* services, at least with the use case specifications and service queries in the experiment. The result suggests that service queries from user task knowledge only was not the most effective service discovery strategy. For instance, two services, *Mobile7Navigation* and *MS2000PortableNavigation*, that were judged as relevant with regards to the journey planning use case contain terms from the original use case description (e.g. *destination*) as well as terms from the *request route* user task model (e.g. *position*). The combination of such terms increased the number of matched terms and therefore the overall service match value.

The partial acceptance of hypothesis **H3** demonstrates that the addition of user task knowledge to service queries led to the retrieval of more *relevant* services. However if TEDDiE and EDDiE both also added domain knowledge using *term expansion* with online thesauri, then the positive effect of adding user task knowledge disappeared: adding domain knowledge had a similar effect to adding user task knowledge. In conclusion adding user task knowledge was not as effective as hypothesised.

In contrast, the rejection of hypothesis **H2** indicates that introducing user task knowledge as defined in TEDDiE did not decrease the numbers of *irrelevant* services retrieved. Indeed, results revealed that adding user task knowledge to domain knowledge in service queries decreased the number of retrieved *relevant* services significantly. Such an inverse relation between precision and recall is not uncommon and is one possible explanation for the low precision score – previous studies have reported on a tendency for precision to decrease as recall increases [52], [53]. Another possible explanation is that the user task models in the catalogue were poorly specified. After all, no independent validation of the models had been undertaken prior to the evaluations. Possible improvements to the specification of the use task models are discussed at the end of the paper.

Finally our partial acceptance of hypothesis **H4** – TEDDiE is more effective than EDDiE without query expansion – revealed that adding user task knowledge to service queries improved effectiveness. Although the mean numbers of retrieved *relevant* and *irrelevant* services per query both increased, the significant difference in the balanced F-score measure suggests that the benefits of more relevant services can outweigh the costs associated with more irrelevant ones. However again, adding user task knowledge to service queries already expanded with domain knowledge from online thesauri did not improve effectiveness further. Indeed, combining terms from user task knowledge and domain knowledge in the same service queries slightly reduced the mean of the balanced F-score measures. The results suggest effective strategies should expand service queries with user task knowledge or domain knowledge, but not both. In the discussion section we explore which can be more effective.

6.5 Threats to Validity

The reported results were collected using controlled experiments in a laboratory setting – a setting that creates possible threats to the validity of the results. This section reports different conclusion, internal, external and construct validity threats [54]. Each is reported in turn.

Threats to construct validity. Construst validity concerns generalizing the results from the reported study to the concept or theory behind the study, namely the application of user task knowledge codified in models during design-time service discovery [54]. One obvious threat to construct validity was the small number of user task models designed for the experiment. The catalogue of 7 rather than 70 user task models probably increased the likelihood that TEDDiE would retrieve relevant user task models and reformulate service queries that retrieve relevant services. A related threat was the small number of service queries used to generate the experimental results. Clearly more studies are needed. That said, extending service query terms from natural language text makes the approach compatible with most industrial requirements and development approaches from the Rational Unified Process [55] to agile [56].

However, in our opinion, the largest threat to construct validity was the quality of the specification of the user task models, and in particular the description of service classes associated to *application* sub-tasks. These descriptions were pivotal to effective service query reformulation and retrieval. In the experiments we adopted a relatively weak approach to describe service classes – simply reusing unaltered text from publicly-available encyclopedias. We believe that a more thorough description of service classes in user task models would probably increase rather than decreaee the precision, recall and balanced F-score measures that TEDDiE would have achieved. The results reported in this paper are probably a minimum level of success with respect to what is possible with the reuse of user task models.

The choice of domain was another potential threat to internal validity. The e-government domain was chosen due its familiarity to the authors as well as the relative availability of domain experts with which to determine service relevance. At this time we cannot make strong claims about the applicability of results to other, less well-understood domains with less repeatability in user tasks and the potential to invoke more complex software services. Repeat studies in other domains are needed.

Threats to internal validity. Threats to the internal validity of the study were influences that could have affected independent variables related to causality [54]. One potential threat was experimenter bias in specifying the use cases, service queries and user task models. One strategy used to mitigate against the threat was to reuse material generated in previous projects without knowledge of the research hypotheses and method applied in the experiment. To this end TEDDiE searched a version of a service registry populated in 2006 as part of the earlier SeCSE integrated project [44]. Likewise we reused use case specifications and service queries from earlier evaluations of EDDiE undertaken before the user task catalogue was devised and developed [35]. This also offset another important internal validity threat - that the terms used to describe the use case specifications were biased to be the similar as the terms used to describe service classes. Both came from different pre-existing sources.

Threats to external validity. Threats to the external validity of the study were influences that could have affected independent variables related to causality [54]. One external validity threat was use of EDDiE as a baseline algorithm and technology against which to compare TEDDiE. A counter-argument runs that EDDiE is a sophisticated service retrieval algorithm that set the bar too high for TEDDiE to succeed. However, since TEDDiE was built on top of EDDiE we conjecture that this comparison is reasonable at least for a first version of evaluation studies.

A related threat was the use of just one technology platform – results might not generalize to other service discovery platforms. Resource constraints meant that it was not possible to carry out directly comparative experiments across several technological platforms. However EDDIE-Lite's implementation of simple keyword matching based on terms extracted directly from natural language text is similar to [42], [43], so we conjecture that the basic findings would still hold with other keyword-based service discovery algorithms and engines. Likewise, whilst detailed natural language descriptions of services are not common, we believe that our use of basic natural language processing algorithms to extract keywords with which to build service queries closely replicates current keyword extraction methods [42], [43].

Threats to conclusion validity. Threats to conclusion validity were concerned with issues that affected the ability to draw correct conclusions about the relations between the treatment and outcome [54]. One important conclusion validity threat was the use of human expertise too determine the sets of relevant and irrelevant services upon which the precision, recall and balanced F-score measures were computed. This expertise was subject to potential biases and inadequate preoperational explication. However, we sought to avoid and mitigate against these biases through careful experimental design. The relevant and irrelevant service sets were created by six judges who worked independently to avoid cross-judge bias. We also minimized bias due to lack of expertise by setting two criteria in selecting each judge. The first was a minimum level of experience with service-oriented computing, to ensure that the roles of candidate services would be fully understood and indicated in the resulting service sets. The second was a minimum level of practical experience with eGovernment web services.

8. FUTURE WORK

In the research reported in this paper we conjectured that reusing knowledge about user tasks codified in models arising from human-computer interaction research would improve the discovery of services during design activities. We learned that reformulating service queries with terms from matched user task models improved service discovery over existing keyword matching, but discovery performance was broadly equivalent to at least one sophisticated thesaurus-based matching technique. This is a potential limitation of such reuse. Given that generating and codifying user task knowledge in models has a cost associated with it not incurred with available on-line thesauri, and that service queries are not restricted to tasks that instantiate the class-level user task models in a catalogue, the reuse of user task models needs to offer more capabilities than implemented in TEDDiE. Let us end the paper with a brief exploration of what these capabilities might be.
The current implementation of TEDDiE associated simple text descriptions of classes of service to invoke during *application* sub-tasks – text from which reformulated query terms were extracted. Therefore one obvious possible extension is a more thorough specification of the terms with which to describe service classes based on a more rigorous analysis of current service specifications drawn from existing service registries. Furthermore we can use wider sources of feature descriptions, such as app stores, to ensure a more systematic and thorough specification of user task models. A second possible extension is to use more formal languages and ontologies with which to describe service classes. Ontologies such as the recent Open Group Service Oriented Architecture (SOA) Ontology [57] can be used to provide both the definitions and a common set of terms for the service classes, but also describe the relationships between those classes. A third possible extension is to exploit other user task model semantics, such as resource descriptions to enrich service queries and temporal associations between sub-tasks. This latter information can provide important guidance not only for service discovery but also for orchestration and choreography based on the required and permitted orderings of service invocation automatically specified with languages such as BPMN version 2.0 [58]. Therefore our research aims to explore the potential of reusing codified user task knowledge describing the problem domain to enhance service-based business process design.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

[1] L.M. Daniele, E. Silva, L.F. Pires, M. Sinderen, R. Poler, R. Sanchis, "A SOA-Based Platform-Specific Framework for Context-Aware Mobile Applications", Enterprise Interoperability, Springer Berlin Heidelberg, pp. 25-37, 2009
[2] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. Di Nitto, V. Mazza, "A context-driven adaptation process for service-based applications", *Proceedings of 2nd International Workshop on Principles of Engineering Service-Oriented Systems. ACM New York*, 2010
[3] A. Marconi, M. Pistore, A. Sirbu, H. Eberle, F. Leymann, and T. Unger, "Enabling Adaptation of Pervasive Flows", *Built-in Contextual Adaptation Lecture Notes in Computer Science*, Volume 5900/2009, pp. 445-454, Springer Berlin / Heidelberg, 2009
[4] D. Diaper, "Understanding Task Analysis for Human Computer Interaction", *D. Diaper and N. Stanton (Eds.), The Handbook of Task Analysis for Human-Computer Interaction*, Lawrence Erlbaum Associates, 2003
[5] F. Paterno, "Task Models in Interactive Software Systems", *Handbook of Software Engineering and Knowledge Engineering*, Volume 1, World Scientific, 817-836, 2002
[6] K. Kritikos, F. Paterno, "Task-Driven Service Discovery and Selection", *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 89-92, 2010
[7] N. Dourdas, X.Zhu, N.A.M Maiden, S. Jones, and K. Zachos, "Discovering Remote Software Service that Satisfy Requirements: Patterns for Query Reformulation", *in Proceedings of CAiSE'06, 18th Conference on Advanced Information System Engineering*, June 5-9, 2006
[8] OASIS Standard, Web Services Business Process Execution Language, Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, 2007

[9] Object Management Group, Business Process Modeling Notation Specification, http://www.omg.org/bpmn/Documents/OMG_Final_Adopted_BPMN_1-0_Spec_06-02-01.pdf, 2006
[10] Adobe, "BPEL4People Overview", http://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html, 2007
[11] J. Preece, Y. Rogers, D. Benyon, S. Holland, and T. Carey, "Human-Computer Interaction", *Addison-Wesley*, 1994
[12] F. Paterno, and C. Santoro "Preventing User Errors by Systematic Analysis of Deviations from the System Task Model", *International Journal of Human-Computer Studies*, Vol. 56, no 2, pp. 225-245, 2002.
[13] G. Mori, F. Paterno and C. Santoro, "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design" *IEEE Transactions on Software Engineering*, vol. 28, no 8, pp. 797-813, 2002.
[14] D. Diaper and N. Stanton, "The Handbook of Task Analysis for Human-computer Interaction", *Lawrence Erlbaum Associates*, 2003.
[15] N. Stanton, "Hierarchical task analysis: Developments, Applications and Extensions", *Applied Ergonomics* 37 (1), 55–79, 2006
[16] B.E. John and D.E. Kieras, "The GOMS family of analysis techniques: Tools for design and evaluation", *Technical Report CMU-CS-94-181, Carnegie-Mellon University*, 1994
[17] R.E. Clark, D.F. Feldon, J.J.G. Van Merrienboer, K.A. Yates, and S. Early, "Cognitive task analysis", *Handbook of research on educational communications and technology*, 3rd ed., pp. 577–593, 2008
[18] F. Paterno, C. Mancini, and S. Meniconi, "ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models", *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*, pp. 362-369, 1997
[19] D. L. Scapin, and J.M.C. Bastien, "Analyse des taches et aide ergonomique a la conception: l'approche MAD*, *C. Kolski (Ed.), Analyse et conception de l'IHM: Interaction homme-machine pour les systemes d'information*, Vol. 1; pp. 85–116, Edition Hermes, 2001
[20] M. van Welie, G. van der Veer, and A. Koster, "An Integrated Representations for Task Modelling", *Proceedings of the Tenth European Conference on Cognitive Ergonomics*, pp. 129-138, 2000
[21] B. Kirwan and L.K. Ainsworth, "A Guide to Task Analysis", *London: Taylor and Francis*, 1992
[22] P. Johnson, H. Johnson, R. Waddington, and A. Shouls, "Task related knowledge structures: Analysis, modelling and application", *Jones, D.M., Winder, R. (Eds.). From Research to Implementation, People and Computers, IV. Cambridge University Press*, Cambridge, UK, 1988
[23] H. Johnson and P. Johnson, "Task Knowledge Structures: Psychological basis and integration into system design", *Acta Psychologica*, 78: 3-26, 1991
[24] F. Paterno, C. Mancini, and S. Meniconi, "ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models," *Proceedings of IFIP TC13 International Conference on HCI*, pp. 362-369, 1997
[25] F. Paterno and C. Mancini "Model-Based Design and Evaluation of Interactive Applications", *Springer-Verlag, London, UK*, 1999
[26] L. Francisco-Revilla and F.M. Shipman, "Adaptive Medical Information Delivery Combining User, Task and Situation Models", *Proceedings of the 5th International Conference on Intelligent User Interfaces (IUI'00)*, 2000
[27] D. Costa and C. Duarte, "Self-adapting TV Based Applications", *Proceedings of the 14th International Conference on HCI*, 2011
[28] F. Paterno, C. Santoro, and L.D. Spano, "User Task-based Development of Multi-device Service-oriented Applications", http://cslab.dico.unimi.it/EUD4Services/papers/EUD4Services-Paterno-et-al-paper.pdf, 2009
[29] F. Paterno, C. Santoro, and L.D. Spano, "ConcurTaskTrees and MARIA languages for Authoring Service-based Applications," http://www.w3.org/2010/02/mbui/soi/paterno-1.pdf, 2010
[30] M. Ruiz, V. Pelechano, and O. Pastor, "Designing Web Services for Supporting User Tasks: A Model Driven Approach" *CoSS International Workshop on Conceptual Modeling of Service-Oriented Software Systems* pp. 193-202, 2006
[31] B.J. Wielinga, "KADS: A modeling approach to knowledge engineering", *Journal of Knowledge Acquisition*, vol.4, no.1, pp.5-53, 1992
[32] U. Keller, R. Lara, and A. Polleres, "WSMO Discovery", *WSMO Working Draft D5.1v0.1.*, http://www.wsmo.org/2004/d5/d5.1/v0.1/, 2004
[33] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel, "A logical framework for web service discovery", *Workshop on Semantic Web Services at ISWC*, 2004
[34] R. Ladner, "Soft computing techniques for web service brokering", *Soft Computing*, 12: 1089-1098, 2008
[35] K. Zachos, N.A.M. Maiden, S. Jones, and X. Zhu, "Discovering Web Services To Specify More Complete System Requirements," *Proc. 19th Conference on Advanced Information System Engineering*, pp.142-157, 2007
[36] K. Zachos, N.A.M. Maiden, and R. Howells-Morris, "Discovering Web Services to Improve Requirements Specifications: Does It Help?" *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08)*, pp. 168-182, 2008
[37] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema," http://www.w3.org/TR/sawsdl/, 2007
[38] J. Kopecky, T. Vitvar, and D. Fensel, "MicroWSMO: Semantic Description of RESTful Services," http://wsmo.org/TR/d38/v0.1/20080219/d38v01 20080219.pdf, 2008
[39] H. Lausen, A. Polleres, and D. Roman, "Web Service Modeling Ontology (WSMO)", *World Wide Web Consortium*, 2005
[40] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services". *W3C Member Submission*, http://www.w3.org/Submission/2004/07, 2004
[41] S. V. Jones, N.A.M. Maiden, K. Zachos, and X. Zhu, "How Service-Centric Systems Change the Requirements Process". *Proceedings REFSQ'2005 Workshop, in conjunction with CaiSE'2005*, pp 105-119, 2005
[42] M. Klein, and A. Bernstein, "Toward high-precision service retrieval," *Internet Computing, IEEE*, vol. 8, no. 1, pp. 30–36, 2004
[43] D. Bachlechner, K. Siorpaes, H. Lausen, and D. Fensel, "Web service discovery a reality check", *3rd European Semantic Web Conference*, 2006
[44] J. Walkerdine, J. Hutchinson, P. Sawyer, G. Dobson, and V. Onditi, "A Faceted Approach to Service Specification," *Proceedings 2nd Int'l Conf. on Internet and Web Applications and Services (ICIW 2007)*, 2007

[45] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to wordnet: An On-line lexical database", *Journal of Lexicography*, 3(4):235-244, 1990
[46] D. Roman, H. Lausen, and U. Keller, "Web service modeling ontology standard (WSMO-standard)", *Working Draft D2v1.0, WSMO*. http://www.wsmo.org/2004/d2/v1.0/, 2004
[47] Wikipedia, "The free Encyclopedia", http://en.wikipedia.org/wiki/Main_Page, February 2011
[48] Encyclopaedia Britannica, "Britannica Online Encyclopedia", http://www.britannica.com/, February 2011
[49] Webopedia, "Online Computer Dictionary for Computer and Internet Terms and Definitions", http://www.webopedia.com/ , February 2011.
[50] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", *New York: ACM Press, Addison-Wesley*, pp. 75ff, ISBN 0-201-39829-X, 1999
[51] G. Salton and M.J. McGill, "Introduction to Modern Information Retrieval", *McGraw Hill Book Co.*, New York, 1983
[52] M. Buckland and F. Gey, "The Relationship between Recall and Precision," *Journal of the American Society for Information Science* vol 45, no 1, pp. 12-19, 1994
[53] L. Egghe, "The Measures Precision, Recall, Fallout and Miss as a Function of the Number of Retrieved Documents and their Mutual Interrelations" *Information Processing & Management*, vol 44, no 2, pp. 856-876, 2008
[54] Wohlin C. "Experimentation in software engineering. An Introduction". *Kluwer Ac. P*, 2000
[55] P. Krutchen, "The Rational Unified Process: An Introduction", *Addison-Wesley*, 2003
[56] A. Cockburn, "Agile Software Development: The Cooperative Game", *Addison-Wesley*, 2006
[57] The Open Group Service Oriented Architecture Ontology, http://www.opengroup.org/bookstore/catalog/c104.htm
[58] BPMN, http://www.omg.org/spec/BPMN/2.0