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.

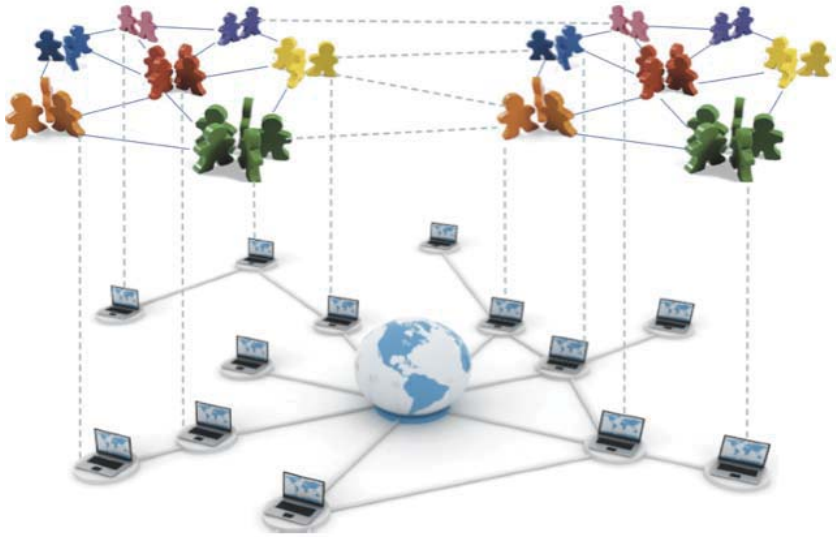

Figure 1. ASNs for GSD

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).

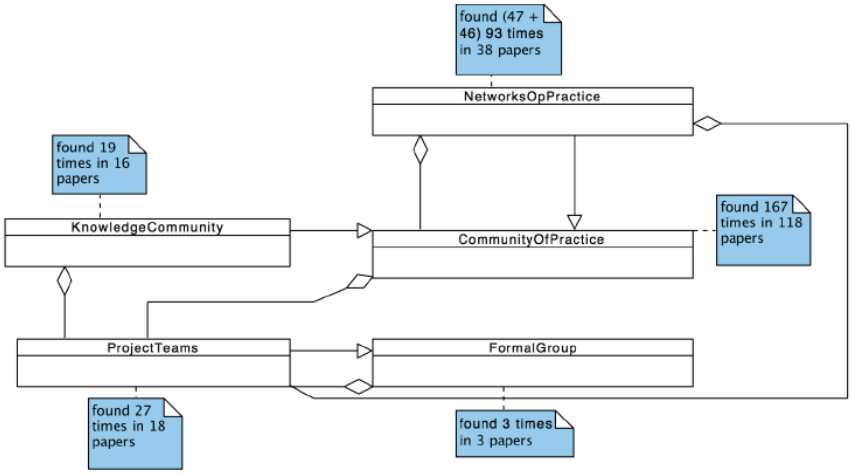

Figure 2. OSS types: project team aggregates

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.

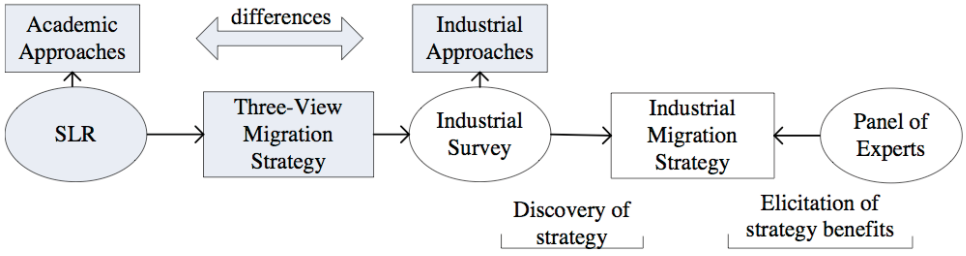

Figure 3. Research context

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.

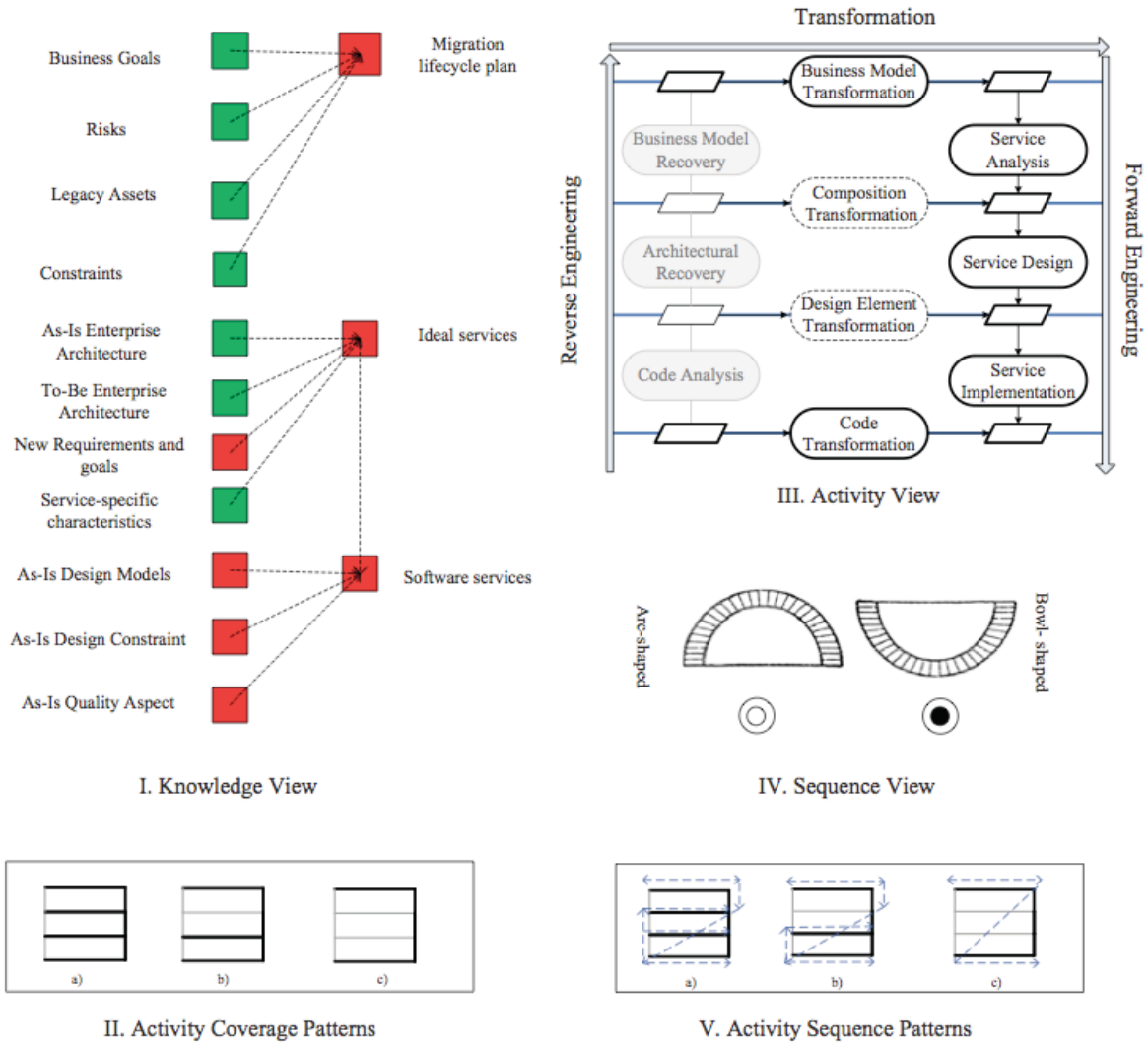

Figure 4. Three view Strategy Representation.

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.