Grant Agreement N° 215483

Title:	*SBA engineering principles comprehensively exploiting HCI and contextual knowledge*
Authors:	*CITY, FBK, Lero-UL, POLIMI, Tilburg, UCBL, UniDue, VUA, TUV*
Editors:	*Elisabetta Di Nitto (Polimi)*
Reviewers:	*Marco Autili (University of L'Aquila), Domenico Bianculli (University of Lugano), Clarissa Marquezan (UniDue), Andreas Metzger (UniDue)*
Identifier:	*Deliverable # CD-JRA-1.1.6*
Type:	*Deliverable*
Version:	*1.0*
Date:	*31 Mar 2011*
Status:	*Final*
Class:	*External*

Management Summary

In this deliverable, we present the research that has been performed in the last year around the S-Cube life-cycle and some related service engineering aspects. Some of the presented contributions validate and extend the S-Cube life-cycle and integrate it with new knowledge about user tasks, inherited from the HCI literature, and context. A special attention is also posed to the problem of service evolution. The various facets of this evolution are considered and a formal framework that allows a definition of service compatibility is provided. A formal framework is also used as the basis for an approach that supports the design and runtime self-adaptation of SBAs. As one of the main goals of industry is to reuse their assets in order not to lose their investments, an important issue for the practical adoption of SBAs is to understand how migration from traditional to SBA systems can be achieved. In this deliverable we present a first step toward a systematization of this field.

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	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
VU Amsterdam	Netherlands

Published S-CUBE documents

These documents are all available from the project website located at http://www.s-cube-network.eu/results/deliverables

Contents

Chapter 1

Introduction

1.1 Overview of the deliverable

In this deliverable, we present the research that has been performed in the last year around the S-Cube life-cycle and some related service engineering aspects. The S-Cube life-cycle [1] represents the main result achieved by the workpackage during the first project year and is one of the assets of the S-Cube Integration Framework [2]. As such, it provides a framework to integrate in a coherent manner the methods and techniques developed by the various other workpackages in the project. Given its importance for the project, its development, refinement and validation is of paramount importance for the JRA-1.1 workpackage (WP).

In this deliverable, the life-cycle has been extended with activities specifically devoted to support pro-active adaptation (see Section 2.1). Moreover, it has been experimented on a concrete case study that has involved a new S-Cube associated partner, which has adopted it for the first time (see Section 2.2). Finally, a survey has been conducted in industry with the aim of understanding how practitioners perceive a life-cycle for Service-Based Applications (SBAs). The result of this survey has lead to the definition and elaboration of a process model that is in line with the S-Cube life-cycle and suggests some minor modifications to it. This experience can be seen as an initial validation of the life-cycle on an industrial setting (see Section 2.3).

One of the most relevant characteristics of SBAs is the ability to adapt to changes in the execution and user contexts. This deliverable also contributes to the research in this area by offering specific design and programming approaches for self-adaptation, as described in Sections 3.1, 3.2 and 3.3. The approach presented in Section 3.2 is also particularly interesting as it refers to SBAs that incorporate *Human-Provided Services*, and that, therefore, are even more subject to changes. All the three approaches take into account the possibility to reason about the *system context*, as changes in this context are some of the main reasons driving self-adaptation.

Adaptation can be seen as a special case of *evolution*, where, usually, this last term is adopted when changes are not applied temporarily and on the fly, but have a permanent impact on the system. In this deliverable the various facets of this evolution are considered, and a formal framework that supports a definition of *service compatibility* is provided (see Section 3.4).

The coherent integration of HCI knowledge in service engineering is another primary challenge for the workpackage. Thus, we also report the result achieved in the adoption of knowledge about users' tasks to drive requirement engineering, design of SBAs and, in particular, service discovery or identification (see Section 4.1). A literature review on the existing methods for service identification is reported in Section 4.2. The review helps in classifying the various approaches and supports designers in the adoption of the approach most suited to their needs.

All aspects mentioned above have been driving the research in JRA-1.1 since the very beginning of the project and are considered to be challenging research problems. Recently, we have been also focusing on identifying issues and challenges of service engineering that are relevant to industry. One

of the aspects we have identified concerns the definition of the modalities for migrating legacy systems into a service-oriented framework. As one of the main goals of industry is to reuse its assets in order not to lose investments, understanding how such migration can take place is certainly an important issue for the practical adoption of SBAs. In this deliverable we present a first step toward a systematization of this field (see Section 4.3).

As in previous deliverables, the research work is shortly summarized in this deliverable and has been reported in greater detail in companion papers. Some of these have been already published while others are under revision at this time. Each work (that in some cases corresponds to more than one paper) is summarized in a different section. Table 1.1 shows the correspondence between the sections, the various topics and the corresponding papers. It can be used as a map to drive the reader through this deliverable. As one of the goals of this deliverable is to integrate contextual and HCI knowledge in a service engineering discipline, Table 1.2 shows which of the presented contributions are related to context and HCI.

The deliverable is structured as follows. Section 1.2 shortly describes the relationships with the other S-Cube workpackages. Section 1.3 provide some background information about the life-cycle in order to ensure the self-containment of this deliverable. Chapters from 2 to 4 present the research contributions based on the structure defined in Table 1.1. Finally, Chapter 5 concludes the report by describing how the presented contributions fit into the challenges defined for the workpackage.

Table 1.1: Map of the research contributions summarized in this deliverable.

Main topic	**Research title**	**Sect**	**Refs**
S-Cube life-cycle	Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle	2.1	[3]
	Using the S-Cube life-cycle for Developing and Running Highly Interactive Distributed applications	2.2	[4]
	A Process Reference Model for Developing Adaptable Service-Based Applications	2.3	[5]
Adaptation and Evolution	Supporting SBA synthesis and adaptation through Service Tiles	3.1	[6]
	A Programming Model for Self-Adaptive Open Enterprise Systems	3.2	[7]
	Towards Context-driven Business Process Adaptation	3.3	[8]
	Supporting Service Evolution	3.4	[9, 10]
Service discovery	Discovering services that facilitate current user tasks	4.1	[11]
	Service Identification Methods: A Systematic Literature Review	4.2	[12]
SOA migration	Surveying the state of the art on service identification and SOA migration	4.3	[13, 14]

1.2 Relationships with the other workpackages

The results presented in this deliverable are strictly intertwined with those in the other research workpackages:

- The work presented in Section 3.3 contributes to JRA-1.2 for what concerns the specific adaptation mechanisms that have been defined, while it contributes to JRA-1.1 for what concerns the design for adaptation guidelines that are offered. This work is also related to the JRA-2.1 WP as it focuses on business processes.

Table 1.2: Map of the contributions related to HCI and context.

Subject	Research title	Sect	Refs
HCI	A Programming Model for Self-Adaptive Open Enterprise Systems	3.2	[7]
	Discovering services that facilitate current user tasks	4.1	[11]
Context	Supporting SBA synthesis and adaptation through Service Tiles	3.1	[6]
	A Programming Model for Self-Adaptive Open Enterprise Systems	3.2	[7]
	Towards Context-driven Business Process Adaptation	3.3	[8]

- The work presented in Section 2.1 is shared with JRA-1.3 and contributes to this last WP for what concerns the specific Quality Assurance techniques being adopted.
- The specific design and programming techniques described in Sections 3.1 and 3.2 can contribute to the JRA-2.2 WP. Moreover, part of the work in Section 3.1 is also related to the JRA-2.3 WP as it is focusing on achieving conversational compatibility between interacting services.
- Finally, the adoption of the life-cycle in practice as presented in Section 2.2 is related to WP IA-3.2 that collects and classifies under the life-cycle all usage scenarios of the S-Cube results.

1.3 The S-Cube life-cycle

The S-Cube life-cycle has the purpose to highlight the importance of adaptation and evolution for service-based applications. It consists of two cycles (see Figure 1.1). In cycle 1, the evolution cycle, shown on the right hand side of the diagram, service-oriented systems are developed following the traditional stages of requirements engineering, design, construction and deployment, while also focusing on quality assurance. In the second cycle, on the left hand side of the figure, systems adapt to changes that happen during their operation.

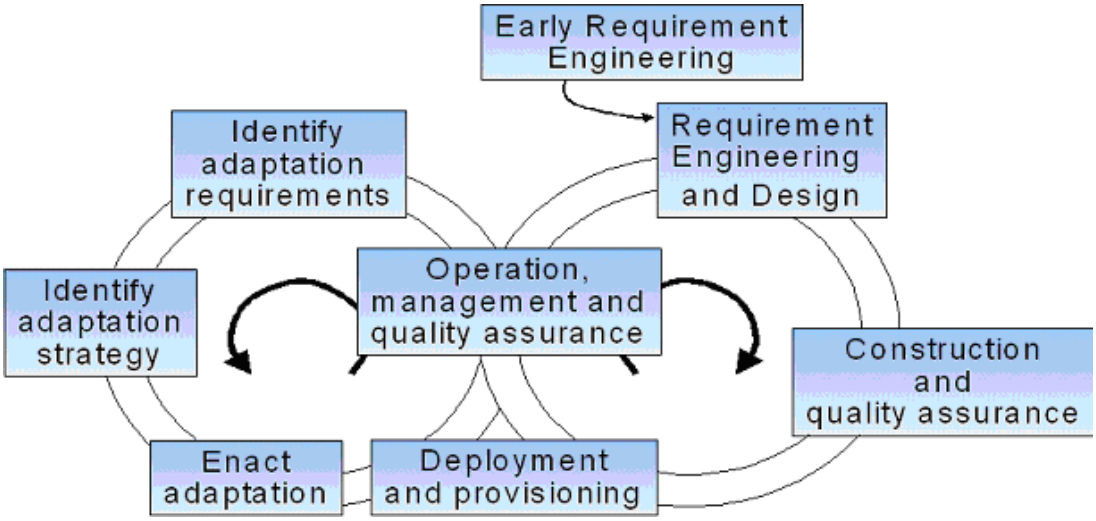

Figure 1.1: The Life-Cycle of Adaptable Service Based Applications.

The two cycles do not conflict with each other. Instead, they coexist and support each other during the lifetime of the application. With services development, adaptation of the system is important to,

or even essential in, system development and design. It is therefore incumbent upon us, in this era of services development, to ensure that software engineering processes are modified to allow the principles of software services development to be enacted successfully during the product life-cycle.

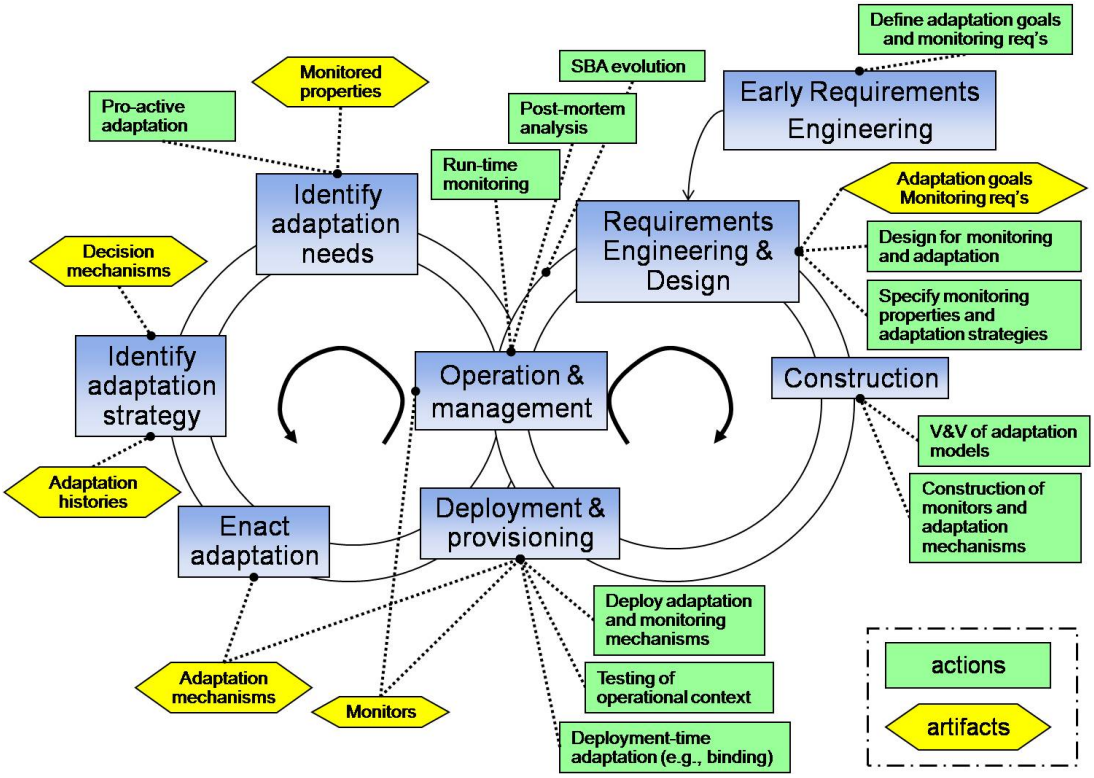

Figure 1.2: Actions and Artifacts within S-Cube life-cycle

In a further development of the life-cycle shown in Figure 1.2 [15], we have identified the various adaptation- and monitoring-specific actions carried out throughout the life-cycle of the Service-Based Application (SBA), the main design artifacts that are exploited to perform adaptation, and the phases where they are used (dotted lines). In the requirements engineering and design phase, the adaptation and monitoring requirements are used to perform the design for adaptation and monitoring. During SBA construction, together with the construction of the SBA, the corresponding monitors and the adaptation mechanisms are being realized. The deployment phase also involves the activities related to adaptation and monitoring: deployment of the adaptation and monitoring mechanisms and deployment time adaptation actions (e.g., binding). During the operation and management phase, the run-time monitoring is executed, using some designed properties, and help the SBA to detect relevant context and system changes. After this phase the left-side of the life-cycle is executed. Here, we can proceed in two different directions, executing evolution or adaptation of the SBA. In the first case we re-start the right-side of the cycle with the requirements engineering and design phase while in the second case we proceed identifying adaptation needs that can be triggered from monitored events, adaptation requirements or context conditions. For each adaptation need it is possible to define a set of suitable strategies. Each adaptation strategy can be characterized by its complexity and its functional and non functional properties.

Chapter 2

Exploiting and evaluating the S-Cube life-cycle

2.1 Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle

Adaptive systems adapt automatically and dynamically to changing conditions. While behaviour of non-adaptive systems is only determined by user input, adaptive systems consider additional information about the application and its context. Issues such as failures of constituent services or diversity in network connectivity may require the application to self-adapt without any human intervention or, at least, limiting this intervention as much as possible.

In [3], the usage of techniques enabling *proactive adaptation* is described along the phases of the S-Cube service life-cycle. Techniques for *proactive adaptation* are able to adapt applications before the undesirable situations happen. This avoids costly compensation and repair activities, as well as unnecessary adaptations, which are deemed key shortcomings of current solutions for adaptive service-oriented applications.

The approach proposes to adopt formalization techniques in all phases of development. This enables the possibility of continuously verify the system also during its operation, and to trigger adaptation when needed. During the requirement analysis and design phase, QoS requirements and assumptions should be formalized together with the functional aspects. To this purpose, we adopt the ALBERT [16] language that offers features to specify temporal aspects, thus allowing the analyst to write expressions such as the following: $onEvent(start, IdentifyParkingTicket) \implies Within(onEvent(end, SendeMail), 1250)$. QoS requirements are checked not only before the SBA is put in the operation environment, but also at runtime. In fact, the identification of the cases when these requirements are not anymore fulfilled can lead to the identification of an adaptation need.

During the design phase, the workflow (or control/data flow) is formalized to support automated checks and reduce the risks of late corrections. In particular, we check whether the workflow satisfies the requirements by using the BOGOR [17] model checker, and exploit the BIR (Bogor Input Representation) language to specify the workflow.

During operation and management, a monitoring platform is used to collect data about the SBA. These data are used to check if the assumptions formalized at design time still hold. If not, the system enters into the adaptation cycle, where, as a first step, it assesses if the violation of an assumption actually leads to the need of actuating an adaptation procedure. Some violations, in fact, could be compensated by other events and, therefore, do not necessarily impact negatively on the execution of the system. An example is the case of a service offering a slow response time that could be compensated by the fact that another services invoked in the same workflow is answering faster than planned. The identification of adaptation strategies to apply can be performed in various different ways. In some cases it could be hard

coded in the SBA, in some others, it could be a function of quality factors and could be calculated based on the data acquired during monitoring.

The paper elaborates on the aforementioned issues and instantiate them through an application example taken from the S-Cube eGovernment scenario.

2.2 Using the S-Cube Lifecycle for Developing and Running Highly Interactive Distributed Applications

The S-Cube service life-cycle model tries to address the main topics arising from adaptable service-based applications. In [4], the authors have proposed an industrial-strength case study for the S-Cube life-cycle model within the emerging and challenging area of Real-time Online Interactive Applications (ROIA); such applications include popular and socially important applications as multi-player online computer games, high-performance simulations, e-learning, etc. The challenges arising from the development of ROIA are several and should be taken into account by a life-cycle supporting the development and the operation of such applications: thousands of users connect simultaneously to one application instance and frequently interact with each other, system must adapt to changing loads, maintaining QoS requirements, etc. The ROIA application used as case study for the life-cycle is the one developed within the European edutain@grid project [18]: a service-oriented architecture including a novel RTF (Real- Time Framework) middleware was implemented focusing on the main challenges of ROIA. One of the main features of such applications is, e.g., the capability to adapt to unforeseen workload or to conditions that may happen requiring a change in the execution environment.

In [4] we evaluate how the S-Cube life-cycle Model could be exploited for the applications on top of the edutain@grid architecture. The experiment has shown that the life-cycle is able to help the designer in the identification of suitable adaptation mechanisms and design patterns for the challenging area in which the application is operating. Since the early phase of the life-cycle, adaptation should be taken into account. In particular, it is important to identify the requirements for adaptation. These drive the selection of the adaptation strategies; in particular five adaptation strategies are identified on the top of edutain@grid architecture for ROIA applications:

- *User migration*: users are migrated from an overloaded server to an underutilized server.
- *Zoning*: new zones are added during runtime and assigned to additional game servers to improve scalability. Zones are seen as a way to organize the virtual space in order to allow a high number of users to access the system.
- *Replication*: new game servers are added transparently during runtime in order to increase computational power for highly frequented zones.
- *Instancing*: creates a copy of a zone which is processed by a different server than the original zone.
- *QoS negotiation* with several distributed hosts.

For the design of adaptable SBAs, adaptation strategies must be related to adaptation triggers and to changes in the context application [19]. Adaptation triggers and suitable trigger rules are defined considering all scenarios at runtime in which application requirements may be violated. By the analysis of ROIA domain, adaptation triggers are related to changes in Service Quality, in the computational context and in users' requirements, in unexpected increment of the users' accesses, and in specific users' needs (**SUCH AS, ...**). In the considered architecture, proactive adaptation is planned on the basis of predicted values from the capacity planning service. It could happen that load balancing may anticipate increasing user numbers in the evening hours and request appropriate resources. In this scenario, the adaptation trigger (predicted increase in user numbers) is related to the change in the context (i.e., time period). Identification of adaptation triggers permits to the application designers to identify the variables

to be monitored at runtime and thereby drives the design of the monitoring mechanism. In ROIA, monitored properties include CPU/memory load on hosting resources, the number of concurrent users in an application instance, bandwidth capacity etc.

This experiment has shown that the S-Cube life-cycle enables both the built-in and the abstraction adaptation, but it also addresses the dynamic adaptation for which it is possible to provide mechanisms that select and instantiate adaptation strategies depending on specific triggers and situations. In the experiment the life-cycle has demonstrated to be effective for ROIA applications in which proactive adaptation is mandatory. The adaptation needs that have been identified by applying the guidelines associated to the life-cycle [19] have allowed the development of demanding applications. Future work will include the implementation of the adaptation triggers identified in the paper and the development of design patterns for different adaptation scenarios.

2.3 A Process Reference Model for Developing Adaptable Service-Based Applications

In [5] we present a process reference model for developing SBAs that have the capability to adapt during runtime. The process model is not a complete life-cycle model and focuses specifically on adaptation specific activities. This research builds on previous work [20, 15], where adaptation-related processes and activities have been identified from various literature sources. The process model outlines the adaptation activities that should occur at each phase of the S-Cube reference life-cycle. Furthermore, the paper suggests some changes to the S-Cube reference life-cycle, most notably the addition of a testing process, and the modification of the deployment and provisioning process. Since this model focuses only on adaptation it can be used in conjunction with the other SBA development techniques presented in this deliverable.

The process model was constructed with data collected from interviews with 7 industrial practitioners and an additional industrial case study where 3 interviews were conducted. The interviews were semi-structured, with the respondents being asked to comment on a frame of reference process model that is based on adaptation activities that were identified in the literature. The frame of reference model was sent to participants before the interviews so that they could examine it and think about process details before the interview. During the interview they were asked based on the experience and opinion whether they agreed with the frame of reference and, if not, how it could be improved. Then they were guided through each phase of the frame of reference model and asked which activities, tasks, artifacts and stakeholders would be required to enact each of the process steps of the model. Each of the interviews were recorded so that they could be later analyzed.

After the interviews were completed they were transcribed so that they could be read carefully and annotated with codes and memos as per Miles and Huberman's [21] qualitative data analysis technique. Each transcribed passage of text was checked for process attributes that could added to the process reference model. Once the text was coded the process attributes were extracted and analysed for duplication and disjointedness. A final set of process attributes were identified for each process and assigned to the appropriate life-cycle processes. In total there were 9 sets of adaptation activities identified for various stages of the S-Cube reference life-cycle. Figure 2.1 illustrates the higher level activities that were identified for the process reference model. The lower level tasks, artifacts and stakeholders are listed separately in the paper.

The validity of the process model was demonstrated by comparing it to a similar approach designed for the adaptation of component-based applications. The component-based approach has itself been validated so it provides a good indication of the basic set of activities required for runtime adaptation, albeit using a different type of architecture. The transferability of the model was demonstrated by showing how the process model fits into a SBA development life-cycle. A good fit into an independently developed SBA development life-cycle illustrates that the reference model is useful in contexts independent to the

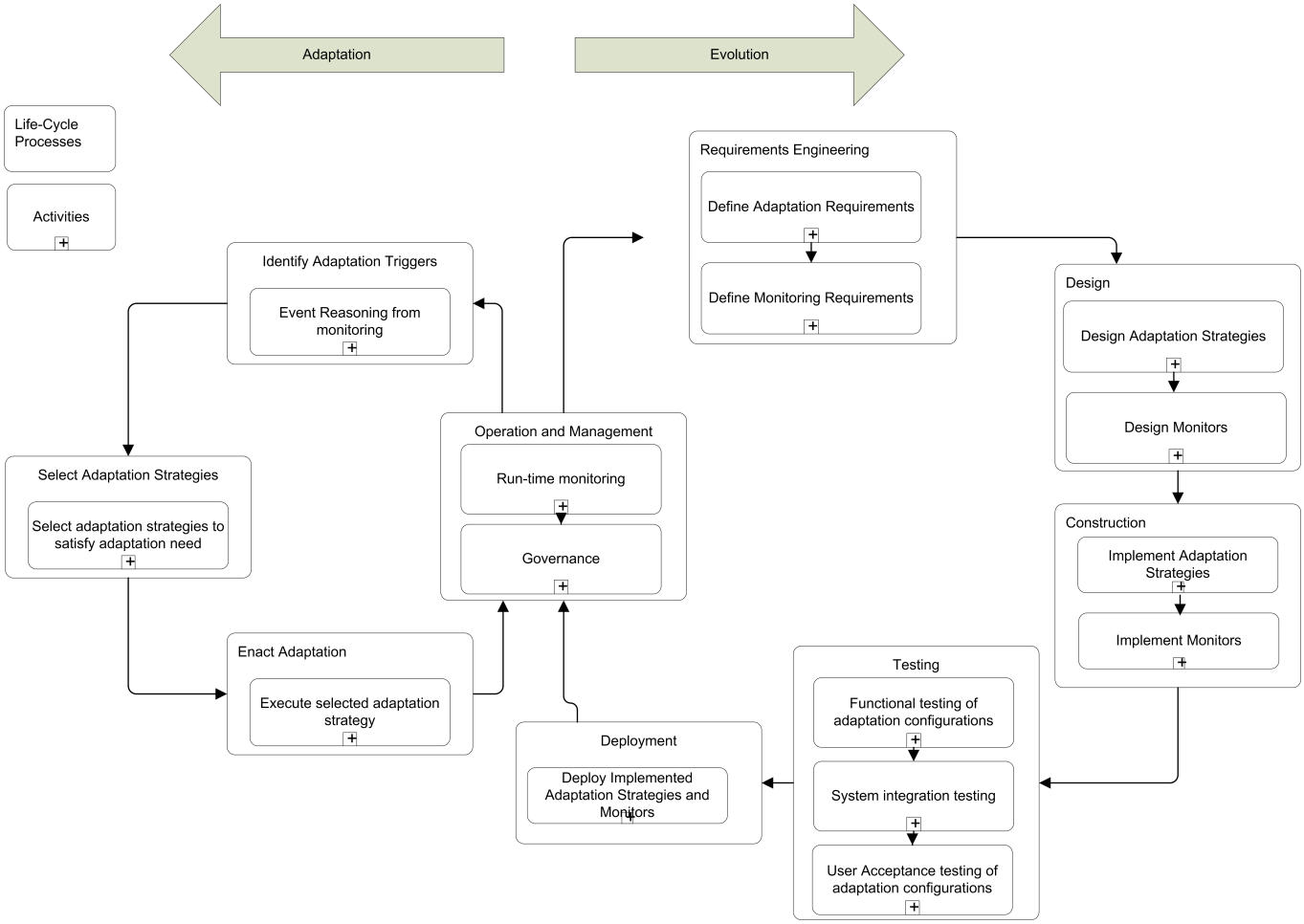

Figure 2.1: V1 - Process Reference Model for Adapting SBAs

context in which it was developed.

The process reference model provides a solid set of guidelines for SBA development practitioners who want to develop adaptable SBAs. It is flexible enough to be used in conjunction with existing life-cycle models lowering its barrier to adoption. An advantage of using this approach is that it is focused on process rather than the adaptation implementation mechanisms.

Chapter 3

Specific approaches to support SBAs evolution and adaptation

3.1 Supporting SBA synthesis and adaptation through Service Tiles

In [6] we propose an approach to support design-time and run-time *self-assembly* of service-based applications. Our design approach makes possible for a developer to express the requirements of an application in intensive terms and to automatically obtain possible design solutions by running a formal, SAT-based machinery. The identified solutions compensate for the limited information often available at design time by changing some decisions at runtime, when the information becomes available. The runtime framework we offer enables the dynamic replacement of the component services of the application, allowing also the integration of previously unforeseen services. In this way the application can self-adapt reacting to changes in its context or to failures of services.

Our work is motivated by the following observations:

- In theory, service-based applications simplify developers' work by composing existing pieces of functionality (the services). However, the available design and implementation tools force designers to focus on specific and low-level aspects, such as the syntactic WSDL interface of the services to be used and the XML structure of the data to be exchanged. Moreover, they assume that designers know about information that are not even explicitly stated in the service interface. An example of such implicit information is the protocol needed to conversate with these services. Mashup approaches (see [22] or [23] for instance) are trying to address such problems but they do not assume that the composition may have dynamic binding capabilities and make the hypothesis that all the services available for composition are known at design time.

- During execution, self-adaptation is mainly supported by so called dynamic binding approaches (e.g., [24, 25]). These, typically, enable a service to be replaced by another service offering exactly the same interface. This limits the set of possible substitutions in a context where conceptually similar services are built by different providers and therefore expose different interfaces.

- Indeed, during execution, those applications that are *context-aware* could need to interact with different services depending on the context in which they or their users are operating. Dynamic binding is the basic building block that could flexibly support this feature. Nevertheless, it has to be complemented by proper mechanisms that allow the service composition to maintain an explicit concept of context, to select proper services based on such context, and to pass such context to the executed services when needed.

In our approach, each *Service* is implemented by one or more *ServiceImplementation*s and it is modeled by a *ServiceUnit* (this can model more than one *Service*). A *ServiceUnit* specifies the requirements

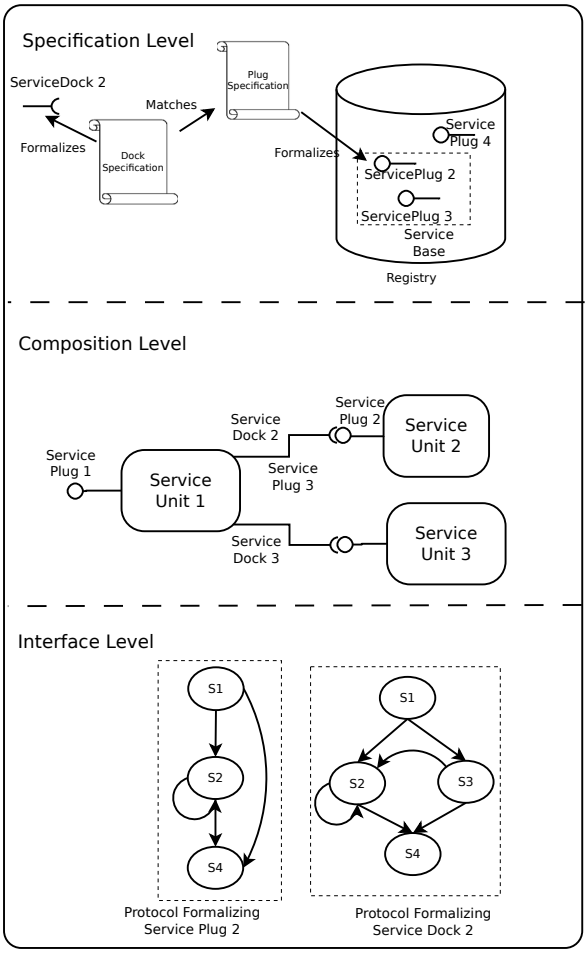

Figure 3.1: Conceptual model of our framework

fulfilled by the corresponding *Services* in terms of *ServicePlug*s, and the requirements delegated to other *Services* in terms of *ServiceDock*s. Both plugs and docks have a specification, represented in Figure 3.1 by the concepts of *PlugSpecification* and *DockSpecification*. Intuitively, a correct *Service Assembly* is constituted by a set of services where each *ServiceDock* in the assembly is bound to a corresponding *ServicePlug*.

The approach works at three levels of abstraction (see Figure 3.1). At the specification level, we assume that services are specified and published. At this same level the designer of a service composition specifies the requirements he/she wants the system to fulfill in terms of one or more *DockSpecification*s. Given such specifications, the registry is searched for identifying matching *PlugSpecifications*. Starting from these last ones, an assembly is built at the composition level, where docks are actually bound to plugs also depending on context and cost information. Such a process is automated and is able to address also the cases where the bound services have themselves specific requirements to be fulfilled by other services. At the communication level, docks and plugs are checked to see if they actually match from the syntactic point of view. If not, proper mapping scripts are automatically generated, starting from the knowledge of the operation and protocol specification. Assemblies and mapping scripts can be recreated/modified at runtime if this is needed to cope with context changes and/or failures/unavailabilities of components.

The machinery for creating assemblies at the composition level is based on the notion of *tile-based system* [26]. Tile-based systems define computations (in our case, service-based applications) as processes that assemble atomic units called *tiles* (that represent services). Each tile can be composed only with certain other tiles, according to the symbols they carry. The resulting assembly process is similar to

building a jigsaw puzzle with the pieces from a given box. The approach shows two interesting characteristics: a) it performs very well by computing service compositions within seconds, b) it can be exploited incrementally, by using parts of solutions computed at one stage as the requirement for computing a new solution in a following stage. This implies that, at runtime, it is possible to run the machinery by providing as requirement the current composition minus the pieces that have to be replaced. Intuitively, the identified solutions compensate for the incorrect or partial information available at design time by changing some decisions at runtime, when the information becomes available.

The ability of using services discovered at runtime and previously unforeseen by the system designer proposes the additional problem of the lack of standardization of their interfaces. In fact newly discovered services may offer interfaces different from those that the composition designer expected to interact with. To cope with this problem, at the communication level, we adapt the interface a client expects to the interface that is actually made available by the selected service. The approach is able to deal with *conversational services* and is based on the synthesis of *mapping scripts*, finite length sequences of instructions, which translate operations that the client is assuming to invoke on the expected service into the corresponding sequences made available by the service that will be actually used. Such mapping scripts are then interpreted by *adapters* that intercept all service requests issued by the client and transform them into the requests the services are able to fulfill.

3.2 A Programming Model for Self-Adaptive Open Enterprise Systems

As interactions and compositions spanning multiple enterprises become increasingly commonplace, organizational boundaries appear to be diminishing in future service-oriented systems. Many service-based applications demand for a mix of interactions between humans and *Software-Based Services (SBS)*. An example is a process model comprising SBS and services provided by human actors also known as *Human-provided Services (HPS)*. Related efforts in service-oriented systems such as WS-HumanTask attempt to model human interactions in top-down business processes assuming closed enterprise systems.

In [7] we discuss service-oriented environments wherein services, and in particular, those provided by humans can be added at any point in time. Following the open world assumption, humans actively shape the availability of HPSs. Such systems may exhibit undesirable properties due to unexpected behavior. Thus, social implications caused by human participation pose additional challenges to designing large-scale mixed SBS-HPS systems. However, with size also the effort for managing dynamically growing and loosely coupled systems is sharply increasing. Periodic adaptations are essential to keep a system within well-defined states, including stable load conditions or desired behavior. Due to the scale and inherent dynamics of open large-scale systems, new approaches for designing, developing and testing HCI in Mixed-Systems are required.

Our approach in this work is bottom-up. We focus on Open Enterprise System (OES) which are closely related to SBAs with a major number of human interactions. The main idea is to observe HCI in service-oriented environments. These observations, with the course of time, allow to actively compensate unexpected behavior of humans, thus keeping the environment in the required boundaries. In the paper we present a programming model that allows to design, create, and deploy service environments that include HPS. It is up to designer to decide whether the system is used as a live system that allows real HCI or as a simulation environment. Our novel *VieCure* framework that allows to actively compensate unexpected behavior and to fine tune the system in any situation. In particular, *VieCure* implements a loop-style log data transformation from events to adaptive actions. One of the major contributions of this work is the policy programming model for *VieCure's* Policy Store. In the presented study, actions are stated in the programming model language of the test environment. The reason is to test the created rules previous to deployment to a real environment.

In conclusion, OESs pose a number of new challenges with their flexibility and their tendency to unexpected behavior, also, because of the human participation. Todays SOA infrastructures can provide the

necessary flexibility, with easily operable interfaces and interaction channels enabling communication and collaboration through HCI for SBA. However, those do not provide means to handle unpredictable changes or system degradations. With humans collaborating and interacting in such networks, there are usually no preplanned top-down composition models. This results in changing interaction and behavior patterns that possibly contradict and at times result in faults from varying conditions and misbehavior in the network.

3.3 Towards Context-driven Business Process Adaptation

The development and procurement of modern business processes, however, have to operate in very dynamic settings. This dynamicity is brought by many different factors: the changes in the realization and behavior of third-party services used in the process, continuously changing properties of the business environment, the factors specific to the underlying technologies and domains (e.g., human factor in case of human-based services). All those context factors normally are out of control of the business process. In order to be robust and competitive, such processes should be able to react at run-time to various changes and situations by adapting its own behavior accordingly. In particular, we are interested in situations where a business process cannot proceed with normal activity execution and fails. In order to recover it at runtime, there is a need for appropriate process adaptation mechanisms. While there exist approaches that achieve adaptability by the use of predefined exception handlers, ECA-rules, or explicitly modeled process variants, in our settings they often fail. Indeed, in a such a volatile context the set of possible situations to be considered at design time is too large and, moreover, unexpected situations still may take place. What we need is a framework, which provides *Context-awareness*. The role of the context is fundamental in realizing the adaptation activities [27] as it enables identifying *when* the process adaptation needed and *what* should be done. Specifically, we need 1) to model the context, 2) to relate services to context changes (to understand in which context they can be executed and how they effect the context), and 3) to observe the current process context. In [8] we present how the context can be used to adapt service-based business processes. The approach that we propose relies on the following key elements:

- *Context-aware model of the application*. We explicitly model the business context, in which the process operates, and describe how the services and policies of the application are related to the context and its changes.
- *Context-aware execution framework*. While executing business process, the process context is continuously monitored in order to check whether it changes as expected by the policies and service specifications.
- *Context-aware adaptation based on automated service composition*. If a critical context change is detected, the adaptation tries to recover the process. This is achieved through construction of a composite service that, starting from the actual context state, performs the necessary changes in the domain to bring the process and its context to the expected state.

At design time, along with the definition of a business process and its implementation on top of a set of available services, relevant contextual information is modeled. First, to model the context properties and their evolution, we define *context property diagrams* that capture possible values of the property (as the diagram states) and the changes of the property values (as transitions). Second, to model how the services change the context, we annotate them with the *effects* on the context properties that correspond to changes of the property value (i.e., to some of the transitions in the property diagram). In other words, the context may change due to the service execution or due to some spontaneous, "unexpected" events. Third, we also capture the business policies over the services to state in which context setting the service may be executed.

We do this by annotating services with the *preconditions* on the context property values. The business context of the reference process is modeled as a set of *context properties*. A context property represents some important characteristic of the environment that can change over time. For example, in a logistics car scenario context properties may be the location of the car, the status of the car (operable/damaged), the status of the treatment area (busy/available), etc. It is important to note that the context property may evolve as an effect of service invocations (e.g., vehicle get repaired after the repair service is engaged), which corresponds to the "normal" behavior of the domain, but also as a result of volatile – "unexpected" – changes.

In these regards we can distinguish *controllable* events, i.e., those that may be achieved through services, and *uncontrollable* events, i.e., exogenous events of the environment. With uncontrollable events we capture unexpected situations that may require process adaptation. The *adaptation strategy* we adopt in this approach is to recover from the process failure so that the process execution can be resumed from the point where it has been blocked. Finally, in order to automatically solve adaptation problems, we use the variation of the service composition approach presented in [28].

3.4 Supporting Service Evolution

Our current efforts on service evolution engineering continues along the lines defined by the [29] and [30] publications, previously reported as part of the JRA-1.1 activities. More specifically, with the term service evolution we denote the continuous process of development of a service through a series of consistent and unambiguous changes [29]. The evolution of a service is expressed through the creation and decommissioning of different service versions during its lifetime. These versions must be aligned with each other in a non-disruptive manner and in a way that allows a service designer to track modifications and their effects on the service. To control service evolution, a designer needs to know why a change was made, what its implications are, and whether the change is consistent. Eliminating spurious results and inconsistencies that occur due to uncontrolled changes is a necessary condition for services to evolve gracefully, ensure stability and handle variability in their behavior.

We classify the nature of service changes depending on their causal effects as:

- Shallow changes: these are small-scale incremental changes that are localized to a service.
- Deep changes: these are large-scale transformational changes cascading beyond the clients of a service, possibly to entire value chains (end-to-end service networks).

With respect to the S-Cube service reference life-cycle, a shallow change to a consumed service (either as part of a composed service or an SBA) does not trigger the need for adaptation on the consumer side. As such, shallow changes ensure that the consumer remains on the right-hand part of the life-cycle. Deep changes to a consumed service on the other hand act as triggers for adaptation, requiring the consumer to react to the observed change and proceed to the left-hand part of the life-cycle. Being able to distinguish between shallow and deep changes is therefore critical for services.

In [9], we proposed a causal model for addressing service changes that deals with the effects of both shallow and deep changes. The article is largely based on concepts and definitions found in [29]. The definitions used in this article have been revised and amended on the basis of formalization and compatibility analysis, prototype implementation, comparison with functionality offered by open standards and an empirical in-depth investigation using an industrial strength case study (the Automotive Purchase Order Processing Scenario, as discussed in [31]).

Shallow changes are initially proposed in [30] and further investigated in [10], in which we present a theoretical framework and language-independent mechanisms to assist service developers in controlling and managing service changes in a uniform and consistent manner. In particular, we provide a sound theory for service compatibility and reasoning mechanisms for delimiting the effect of changes which we keep local to and consistent with a service description. The approach proposed in [10] and discussed at

length in [32] is in contrast to the vast majority of existing approaches in the field. Most such approaches view service compatibility as strictly enforced by a set of empirical and technology-specific rules (e.g., WSDL-dependent guidelines) which indicate which changes are characterized as being compatible. This results in a very strict service evolution regime which prohibits potentially legitimate changes from being applied due to the limitations of current service technologies.

To this effect, in [10] we present a formally-backed compatible service evolution framework which is based on a technology-agnostic notation for the representation of services in the form of Abstract Service Descriptions (ASDs). ASDs act as a springboard to explain the versioning mechanisms for services. Using these results, we formally define service compatibility and develop a theory for the compatible evolution of services. As part of this approach we introduce the notion of *T-shaped changes*, which enforce service compatibility between interrelated service versions in two dimensions: horizontally and vertically. We also demonstrate how to reason about the compatibility of service versions using a Compatibility Checking Algorithm (CCA) in order to decide if changes to them are T-shaped or not. We validate the presented framework in practice by means of a proof-of-concept prototype implementation in the form of the Service Representation Modeler (SRMod) tool and a case study (the Automotive Purchase Order Processing validation scenario, as above). Based on the findings of this validation we provided a series of recommendations for the improvement of service description languages in the context of service evolution.

Chapter 4

Other software engineering aspects around SBAs and SOA

4.1 Discovering services that facilitate current user tasks

The research reported here [11] relates to the Requirements Engineering and Design phase of the service life cycle documented in [2]; it extends efforts reported in [33] to address the perceived lack of integration of user task knowledge in service discovery.

Standard approaches to model services' context of use normally focus on the business process to be executed, but lack important information about the actor performing the process and their goals, actions and constraints (e.g., [34]). We suggest that user task knowledge can improve the discovery and selection of best-fit services by bridging the semantic mismatch that occurs due to the different ontologies used to describe a problem query (as pertaining to a user's goal and tasks) and a solution service. We designed, implemented and evaluated user task models for service discovery using the SeCSE service discovery environment and EDDiE, its associated service discovery engine; we report here the results of our studies testing the following research hypotheses:

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

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

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

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

In order to represent user task knowledge, we described domain tasks covering the e-Government case study described in [35], then manually generated reusable, domain-independent task models using the CTT formalism to describe user tasks as structured, purposeful activities executed during interaction with a system. We further specified in structured natural language the functionality and operations of candidate classes of software services that, based on past experience, support users to achieve specified tasks. Finally, we developed and populated an online task model catalogue linking knowledge about classes of service solution to classes of user tasks. We then extended and implemented the EDDiE service discovery algorithm to perform Task-based service discovery as follows:

1. *User Task model match*: T-EDDiE uses its existing algorithm to match the expanded and disambiguated service query to the description part of each user task model. The result is an ordered set of retrieved user task models that match the service query.

2. *Reformulate service query*: T-EDDiE uses the described classes of service in the solution part of each retrieved user task model to generate one or more new service queries that are expressed in terms of the service features rather than consumer requirements. The analyst can then modify these reformulated queries as needed.

3. *Service match*: the service discovery algorithm uses each revised service queries to discover candidate service specifications in service registries. The result is an ordered set of service specifications that match to the revised service query.

An evaluation was then designed to undertake a summative evaluation of the Task-based algorithm (T-EDDiE) and investigate our four hypotheses using precision, recall and balanced F-score statistical measures.[1] The evaluation was carried out in four stages:

1. Extend user task models with knowledge about classes of software service.

2. Undertake a human assessment of candidate web services relevant to our use case (government journey planning use case).

3. Compare the statistical measures of T-EDDiE's query reformulation strategies (QRS) to determine which strategy is performing best when incorporating user task models in the service discovery process.

4. Compare the statistical measures of T-EDDiE against the original EDDiE algorithm.

Paired t-tests run on T-EDDiE's 4 QRS (*Extend* service query with user task knowledge both *with* and *without* the application of term expansion; *replace* service query with user task knowledge both *with* and *without* the application of term expansion. A more detailed explanation can be found in [11]) showed little significant differences with regards to precision, but provided enough evidence to corroborate our first research hypothesis H1: Extending rather than replacing service queries with additional knowledge about user tasks will *improve the overall effectiveness of services*. Subsequent evaluation activities investigated the performance of both EDDiE and T-EDDiE based on the retrieval of web services judged as relevant with regards to our journey planning use case, with findings as follows:

- H2 (T-EDDiE will *decrease the number of irrelevant services*) was rejected as both precision and mean average precision of T-EDDiE's results were lower than the equivalent scores for EDDiE.

- H3 (T-EDDiE will *increase the number of relevant services*) was accepted due to the fact that T-EDDiE's recall score was indeed significantly higher than EDDiE's recall value.

- H4 (T-EDDiE will *improve the overall correctness of services*) was accepted but only for *non-expanded queries* because of the balanced F-score measure that revealed a superior value for T-EDDiE.

Planned further work will address threats to the validity of our reported results, including the randomness and size of the tasks and service samples selected for this study, the sample size for the second part of the evaluation, and the choice of domain and will generalise our findings outside of government journey planning to other domains.

[1] The balanced F-score assesses the effectiveness of both query mechanisms by identifying the harmonic mean of the precision and recall where recall and precision are evenly weighted.

4.2 Service Identification Methods: A Systematic Literature Review

In the design of services, service identification (SI) is a significant task aiming at determining (based on available resources) services that are appropriate for use in an SOA. Many service identification methods (SIMs) have been proposed from both academia and industry. However, these SIMs differ significantly, ranging from source code extraction to business domain analysis, from bottom-up to top-down strategy and from ontology-based process to guideline-driven process. Accordingly, the inputs and outputs of these approaches vary as well. Some approaches start with business process whereas some others start with domain knowledge (e.g., goals and strategies); some approaches focus on business services whereas others focus on software services.

Given many SIMs, a common question that practitioners face is how to select the most appropriate SIM that copes with the available resources and fits their needs [36]. Some enterprises, for instance, not only have well defined business process models in place but also well documented goals, business partners and enterprise capabilities (e.g., IT resources) to support its business process. For them, a SIM that takes all of this information into account will be more suitable than those identifying services only based on e.g., business processes.

Despite the comparisons (e.g., [37, 38]) presented so far, there is no systematic analysis of the existing SIMs. As a result, a holistic overview of extant SIMs is missing. Moreover, the criteria used in the existing comparison frameworks cover many aspects, including economic, business, and technical aspects. However, a comparison of the basic elements (such as the inputs, outputs and processes) of the SIMs is currently missing. When selecting SIMs one often starts questioning what is required for using certain SIMs, what can be expected from them and how to carry them out, before addressing other requirements. Without such an overview of the basic elements of SIMs, the selection of SIMs becomes more complicated. Accordingly, the following research questions arise:

- *RQ1: What are the existing SIMs?*
- *RQ2: How do the SIMs differ? This can be elaborated into: RQ2.a what types of inputs do the SIMs start from? RQ2.b what types of services do the SIMs produce? RQ2.c what types of strategies and techniques are used by the SIMs?*

To answer these research questions, in [12] we have decided to conduct a systematic literature review. Systematic literature reviews are particularly powerful in collecting and analyzing existing work. Indeed, they can maximize the chance to retrieve complete data sets and minimize the chance of bias by summarizing the existing evidence [39]. An overview of the surveyed SIMs is given in Table 4.1.

The variety in the types of inputs, outputs and techniques that can be inferred by analyzing Table 4.1 explains why practitioners often face difficulties to select a SIM that both fits their needs and copes with the available resources (a more detailed table is presented in the paper). To help compare and select among the SIMs, we have created an input-output matrix, where rows represent the types of outputs produced by the SIMs and columns represent the types of inputs being used. Each cell of the matrix describes a specific SIM if it uses the input and produces the output represented by the column and row respectively. This matrix helps in the selection of SIMs allowing stakeholders in the identification of the method that best suits their needs on the basis of what is going to be produced (the outputs) and the needed resources (the inputs). Moreover, it enables an immediate comparison between alternative methods.

4.3 Surveying the state of the art on service identification and SOA migration

One of the key promises of service oriented paradigm is facilitating reuse of enterprise assets in legacy systems. Migration of legacy systems to service-based systems enables achieving advantages offered by SOA while still reusing the embedded capabilities in the legacy systems.

Table 4.1: An overview of the existing SIMs

SIM	Year	Type of input	Type of output	Technique	Validation
S1 [40]	2004	Application domain	A list of services	Algorithm	Evaluated
S2 [41]	2004	Mix	Formal service specification	Algorithm	Evaluated
S3 [42]	2005	Business process	Service model	Algorithm	No
S4 [43]	2005	Legacy system	Service implementation	Analysis	Case study
S5 [44]	2005	Legacy system	Service implementation	Algorithm	Case study
S6 [45]	2006	Data	Informal service specification	Pattern	No
S7 [46]	2007	Application domain	Informal service specification	Guidelines	Case study
S8 [47]	2007	Application domain	Service model	Algorithm	No
S9 [48]	2007	Application domain	Informal service specification	Guidelines	Project
S10 [49]	2007	Business process	Service model	Guidelines	No
S11 [50]	2007	Use case	Informal service specification	Algorithm	Example
S12 [51]	2007	Business process	Service model	Analysis	Example
S13 [52]	2008	Mix	Service model	Analysis	Case study
S14 [53]	2008	Application domain	Service model	guidelines	Case study
S15 [54]	2008	Business process	Service implementation	Algorithm	Case study
S16 [55]	2008	Business process	Service model	Algorithm	Evaluated
S17 [56]	2008	Business process	A list of services	Algorithm	Example
S18 [57]	2008	Feature	Informal service specification	Analysis	Case study
S19 [58]	2008	Feature	Informal service specification	Ontology	Case study
S20 [59]	2008	Legacy system	Service implementation	Information manipulation	Project
S21 [60]	2008	Mix	Informal service specification	Analysis	Example
S22 [37]	2009	Application domain	Informal service specification	Analysis	No
S23 [61]	2009	Business process	A list of services	Ontology	Case study
S24 [62]	2009	Business process	Service model	Ontology	Case study
S25 [63]	2009	Business process	Informal service specification	Algorithm	Case study
S26 [64]	2009	Business process	Service implementation	Algorithm	Evaluated
S27 [65]	2009	Legacy system	Service implementation	Ontology	Case study
S28 [66]	2009	Use case	Informal service specification	Guidelines	Example
S29 [67]	2009	Data	Informal service specification	Guidelines	Example
S30 [68]	2009	Mix	Service implementation	Guidelines	Case study

Since the early use of SOA, migration of legacy systems to SOA has caught a lot of attention. Various studies present an approach for such migration. These studies mainly differ in the way they provide solutions for two challenging problems of what can be migrated (i.e., the legacy elements) and how the migration is performed (i.e., the migration process). As an example, some studies address implementation aspects of migration by providing methods for altering segments of the legacy code to web services. On the other hand, other studies focus on refactoring the legacy architecture to a service-based architecture based on business drivers such as business rules, benefits and risks.
Given such differences among SOA migration approaches it is hard to achieve a general understanding of *how to perform SOA migration* and consequently it is hard to determine the SOA migration strategy. To define a migration strategy, various aspects such as what activities are needed for such migration, what are the available knowledge assets, and what should drive the whole migration, needs to be considered. Accordingly, to select a migration approach, to be used in the strategy, it is essential to know how those aspects are addressed in that specific approach. A reference that categorizes and characterizes different approaches using the mentioned aspects facilitates systematically determining the migration path to take.

To obtain the SOA Migration categorization, a systematic review that extracts migration categories existing in the field was conducted and reported in [13]. The strength of systematic reviews in minimizing the bias in the review process enhanced the extraction of sound and meaningful categorization of the migration approaches. Such categorization brings order on the existing SOA migration approaches and provides insight on 'how to perform SOA migration'. As part of the systematic process, we developed a protocol that provided a plan for the review in terms of the method to be followed, including the research questions and the data to be extracted. Following this protocol we identified 51 unique primary studies that specifically provide a solution for migration of legacy systems to SOA.
To analyze the primary studies and further to identify the SOA migration categories, we needed to represent the constituent activities of various approaches in a unified manner. To this end, we used a framework called SOA-MF, and mapped all primary studies on it. SOA-MF stems from existing theory on reengineering and architectural recovery and is published in our earlier work [14]. This framework is a skeleton of the holistic migration process along with the distinct conceptual elements involved in such a process. The framework consists of three sub-processes: reverse engineering, transformation and forward engineering (see Figure 4.1.I).

By devising a coding procedure based on conceptual elements of SOA-MF, we analyzed the studies and extracted eight distinct SOA migration families. To identify these families, we mapped each study on SOA-MF and clustered them into different families based on the similarity of their mappings. Figure 4.1.III illustrates the schematic form of distinguished mappings that are dedicated to each family. At first glance, a SOA migration family represents a set of approaches with graphically similar mappings on SOA-MF. Despite their simplicity, the mappings reflect the following information about the migration process: to what extent the reverse engineering, transformation and forward engineering occur, what activities are carried out, what artifacts are used or produced, and what abstraction levels are covered. By positioning a migration approach on these families, insight in the following aspects can be achieved: what is migrated, how the migration is carried out, what is the main objective of migration, and finally, what are the available solutions.

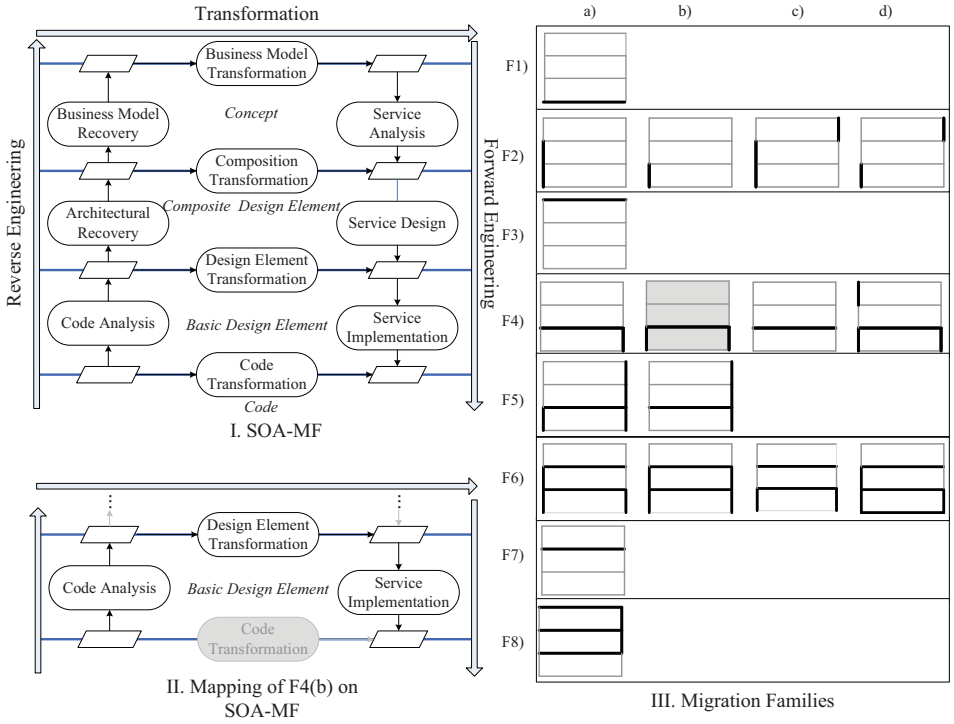

Figure 4.1: SOA Migration Families

Chapter 5

Research Contribution and conclusion

The work presented in this deliverable shows a good step toward the fulfillment of the JRA-1.1 challenges as they have been defined in the Integration Research Framework deliverable [69]. In the following we list the challenges that have been defined for the workpackage (shown in *italics*) and we discuss the extent of their fulfillment.

- *Definition of a coherent life cycle for adaptable and evolvable SBA* and *Measuring, controlling, evaluating and improving the life cycle and the related processes.* With this deliverable, the S-Cube life-cycle has started to be a living element. Not only it has been extended to support pro-active adaptation in Section 2.1, but it has been also experimented for the development of a non-conventional SBA (see Section 2.2) and has been analyzed with respect to practitioner's processes and feedbacks (see Section 2.3). Moreover, the contributions presented in Sections 4.1 and 4.2 contribute especially to the requirement engineering phase by supporting the discovery and identification of services, while the contributions presented in Sections 3.1, 3.2, 3.3 and 3.4 focus mainly in the design and realization phases, where adaptable service-based applications are constructed.

- *HCI and context aspects in the development of service based applications.* The deliverable shows that HCI has an important role both because it supports developers in performing their tasks (see Section 4.1) and because it is the way human-provided services can be integrated with software services in a seamless way (see Section 3.2).

- *Understand when an adaptation requirement should be selected.* Self-adaptable SBAs are able to identify the needs for adaptation while they are running and to select the proper adaptation actions. The contributions presented in Sections from 3.1 to 3.4 are all addressing part of this challenge from different viewpoints. Certainly, more work has to be performed to incorporate all these efforts in a coherent framework.

- *Identify best practices for SOA migration.* This challenge has been recently added to the others defined for the workpackage. The work discussed in Section 4.3 is certainly addressing, at least from a conceptual perspective.

The plan for future work concerns mainly the analysis of the results achieved by the JRA-2 activity to conceptually integrate them within the life-cycle, further refinements and evaluation of the life-cycle, and, in general, a consolidation of the results presented in this deliverable.

Bibliography

[1] "CD-JRA-1.1.1 state of the art report on software engineering design knowledge and survey of hci and contextual knowledge." [Online]. Available: http://www.s-cube-network.eu/

[2] S-Cube Consortium, "Integration framework baseline," S-Cube project, Tech. Rep., 2009.

[3] A. Metzger, E. Schmieders, C. Cappiello, E. Di Nitto, R. Kazhamiakin, B. Pernici, and M. Pistore, "Towards proactive adaptation: A journey along the s-cube service life-cycle," in *MESOA: 4th International Workshop on Maintenance and Evolution of Service-Oriented Systems*, 2010.

[4] D. Meilander, 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*, ser. Lecture Notes in Computer Science, E. Di Nitto and R. Yahyapour, Eds. Springer Berlin / Heidelberg, 2010, vol. 6481, pp. 175–186.

[5] S. Lane, A. Bucchiarone, and I. Richardson, "A process reference model for developing adaptable Service-Based applications," *Submitted for publication*, 2011.

[6] L. Cavallaro, E. D. Nitto, and M. Pradella, "An approach to develop self-assembling self-adaptive service oriented applications," Politecnico di Milano, Tech. Rep., 2011, submitted for publication.

[7] H. Psaier, F. Skopik, D. Schall, L. Juszczyk, M. Treiber, and S. Dustdar, "A programming model for self-adaptive open enterprise systems," in *5th Workshop of the 11th International Middleware Conference 2010 (MW4SOC)*, Bangalore, India, 2010.

[8] A. Bucchiarone, R. Kazhamiakin, M. Pistore, and H. Raik, "Adaptation of service-based business processes by context-aware replanning," in *Submitted to ICWS 2011*, 2011.

[9] M. P. Papazoglou, V. Andrikopoulos, and S. Benbernou, "Managing evolving services," *IEEE Software's SWSI: Component Software beyond Software Programming (to appear)*, Jun. 2011.

[10] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the evolution of services," *IEEE Transactions on Software Engineering (under revision)*, 2011.

[11] K. K. Zachos, A. Kounkou, and N. Maiden. (2009) Exploiting codified user task knowledge to discover services.

[12] Q. Gu and P. Lago, "Service identification methods: A systematic literature review," in *ServiceWave 2010: Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, vol. 6481. Springer Berlin / Heidelberg, 2010, pp. 37–50.

[13] M. Razavian and P. Lago, "A frame of reference for soa migration," in *ServiceWave 2010: Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, vol. 6481. Springer Berlin / Heidelberg, 2010, pp. 150–162.

[14] ——, "Towards a conceptual framework for legacy to soa migration," in *Fifth International Workshop on Engineering Service-Oriented Applications (WESOA'09)*, 2009.

[15] 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 *Fifth international Workshop on Engineering Service-oriented Applications (WESOA)*, Stockholm, Sweden, 2009.

[16] E. Dubois, "Albert: A formal language and its supporting tools for requirements engineering," in *Fundamental Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, E. Astesiano, Ed. Springer Berlin / Heidelberg, 1998, vol. 1382, pp. 322–325.

[17] "BOGOR," bogor.projects.cis.ksu.uk.

[18] "The edutain@grid project," 2009, http://www.edutaingrid.eu.

[19] 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 *ICSOC/ServiceWave Workshops*, 2009, pp. 467–476.

[20] S. Lane, Q. Gu, P. Lago, and I. Richardson, "Adaptation of service based applications: A maintenance process?" Lero, the Irish Software Engineering Research Centre, Limerick, Ireland, Tech. Rep. Lero-TR-2010-08, 2010. [Online]. Available: http://www.lero.ie/sites/default/files/files/Lero-TR-2010-08.pdf

[21] M. B. Miles and A. M. Huberman, *Qualitative data analysis: An expanded sourcebook*. SAGE publications, Inc, 1994.

[22] L. Baresi and S. Guinea, "Mashups with mashlight," in *ICSOC*, 2010, pp. 711–712.

[23] F. Daniel, F. Casati, S. Soi, J. Fox, D. Zancarli, and M.-C. Shan, "Hosted universal integration on the web: The mashart platform," in *ICSOC/ServiceWave*, 2009, pp. 647–648.

[24] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, and Z. Wu, "The METEOR-S approach for configuring and executing dynamic web processes," University of Georgia, Athens, Tech. Rep., June 2005.

[25] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani, "Paws: A framework for executing adaptive web-service processes," *IEEE Software*, vol. 24, no. 6, pp. 39–46, 2007.

[26] D. Giammarresi and A. Restivo, "Two-dimensional languages," in *Handbook of Formal Languages, Vol. 3, Beyond Words*. Springer, 1997.

[27] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in *Proc. HUC*, 1999, pp. 304–307.

[28] P. Bertoli, R. Kazhamiakin, M. Paolucci, M. Pistore, H. Raik, and M. Wagner, "Control Flow Requirements for Automated Service Composition," in *Proc. ICWS'09*, 2009, pp. 17–24.

[29] M. P. Papazoglou, "The challenges of service evolution," in *Proceedings of the 20th international conference on Advanced Information Systems Engineering (CAiSE '08)*. Springer-Verlag, 2008, pp. 1–15.

[30] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "Managing the evolution of service specifications," in *Proceedings of the 20th international conference on Advanced Information Systems Engineering (CAiSE '08)*. Springer-Verlag, 2008, pp. 359–374.

[31] "CD-IA-3.2.1 initial definition of validation scenarios," Oct. 2009. [Online]. Available: http://www.s-cube-network.eu/

[32] V. Andrikopoulos, *A Theory and Model for the Evolution of Software Services*, ser. CentER PhD Dissertation Series. Tilburg, Netherlands: Tilburg University Press, 2010, no. 262.

[33] "CD-JRA-1.1.5 analysis on how to exploit codified hci and codified context knowledge for sba engineering." [Online]. Available: http://www.s-cube-network.eu/

[34] "CD-JRA-1.1.2 separate design knowledge models for software engineering and service based computing." [Online]. Available: http://www.s-cube-network.eu/

[35] "CD-IA-2.2.4 report on common pilot cases." [Online]. Available: http://www.s-cube-network.eu/

[36] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[37] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann, "Identification and analysis of business and software services—a consolidated approach," *IEEE Transactions on Services Computing*, vol. 2, no. 1, pp. 50–64, 2009.

[38] R. Boerner and M. Goeken, "Service identification in SOA governance literature review and implications for a new method," in *Int. Conference on Digital Ecosystems and Technologies*, 2009, pp. 588–593.

[39] B. Kitchenham, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[40] H. K. Jain, H. Zhao, and N. R. Chinta, "A spanning tree based approach to identifying web services," *Int. J. Web Service Res.*, vol. 1, no. 1, pp. 1–20, 2004.

[41] Z. Zhang and H. Yang, "Incubating services in legacy systems for architectural migration," in *Asia-Pacific Soft. Eng. Conf.* IEEE CS, 2004, pp. 196–203.

[42] Z. Wang, X. Xu, and D. Zhan, "Normal forms and normalized design method for business service," in *Int. Conference on e-Business Eng.* IEEE CS, 2005, pp. 79–86.

[43] F. Chen, S. Li, and W. C.-C. Chu, "Feature analysis for service-oriented reengineering," in *Asia-Pacific Soft. Eng. Conference*. IEEE Computer Society, 2005, pp. 201–208.

[44] Z. Zhang, R. Liu, and H. Yang, "Service identification and packaging in service oriented reengineering," in *Int. Conference on Software Engineering and Knowledge Engineering*, 2005, pp. 241–249.

[45] Y. Baghdadi, "Reverse engineering relational databases to identify and specify basic web services with respect to service oriented computing," *Information Systems Frontiers*, vol. 8, no. 5, pp. 395–410, 2006.

[46] K. Klose, R. Knackstedt, and D. Beverungen, "Identification of services - a stakeholder-based approach to SOA development and its application in the area of production planning," in *ECIS*. University of St. Gallen, 2007, pp. 1802–1814.

[47] S. Chaari, F. Biennier, J. Favrel, and C. Benamar, "Towards a service-oriented enterprise based on business components identification," *Enterprise Interoperability II*, pp. 495–506, 2007.

[48] F. Kohlmann and R. Alt, "Business-driven service modeling - a methodological approach from the finance industry," in *Int. Working Conference on Business Process and Services Computing*, 2007.

[49] S. Inaganti and G. K. Behara, "Service identification: BPM and SOA handshake," 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.2377

[50] Y. Kim and K.-G. Doh, "The service modeling process based on use case refactoring," *Business Information Systems*, pp. 108–120, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72035-5_9

[51] J. Amsden, "Modeling SOA: Part 1. service specification," IBM Dev. Works, 2007.

[52] N. Fareghzadeh, "Service identification approach to SOA development," in *World Academy of Science Engineering and Technology*, vol. 35, 2008.

[53] S. Kim, M. Kim, and S. Park, "Service identification using goal and scenario in service oriented architecture," in *APSEC*. IEEE, 2008, pp. 419–426.

[54] S. Mani, V. S. Sinha, N. Sukaviriya, and T. Ramachandra, "Using user interface design to enhance service identification," in *Int. Conference on Web Services*. IEEE, 2008.

[55] P. Jamshidi, M. Sharifi, and S. Mansour, "To establish enterprise service model from enterprise business model," in *Int. Conf on Services Computing*. IEEE, 2008.

[56] V. Dwivedi and N. Kulkarni, "A model driven service identification approach for process centric systems," in *Congress on Services*. IEEE CS, 2008, pp. 65–72.

[57] J. Lee, D. Muthig, and M. Naab, "An approach for developing service oriented product lines," in *Int. Software Product Line Conf.* IEEE, 2008, pp. 275–284.

[58] D. Kang, C.-y. Song, and D.-K. Baik, "A method of service identification for product line," in *ICCIT*. IEEE, 2008, pp. 1040–1045.

[59] L. Aversano, L. Cerulo, and C. Palumbo, "Mining candidate web services from legacy code," in *Int. Symposium on Web Site Evolution*, 2008, pp. 37–40.

[60] M. J. Cho, H. R. Choi, H. S. Kim, S. G. Hong, Y. Keceli, and J. Y. Park, "Service identification and modeling for service oriented architecture applications," in *Int. Conference on Software Engineering, Parallel and Distributed Systems*. World Scientific and Engineering Academy and Society, 2008, pp. 193–199.

[61] D. Bianchini, C. Cappiello, V. Antonellis, and B. Pernici, "P2S: A methodology to enable inter-organizational process design through web services," in *Int. Conf. on Advanced Information Systems Engineering*. Springer, 2009, pp. 334–348.

[62] R. Yousef, M. Odeh, D. Coward, and A. Sharieh, "BPAOntoSOA: A generic framework to derive software service oriented models from business process architectures," in *ICADIWT*, 2009, pp. 50–55.

[63] L. G. Azevedo, F. Santoro, F. Baiao, J. Souza, K. Revoredo, V. Pereira, and I. Herlain, *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2009, ch. A Method for Service Identification from Business Process Models in a SOA Approach, pp. 99–112.

[64] Y. Kim and K.-G. Doh, "Formal identification of right-grained services for service-oriented modeling," in *Int. Conference on WISE*. Springer, 2009, pp. 261–273.

[65] F. Chen, Z. Zhang, J. Li, J. Kang, and H. Yang, "Service identification via ontology mapping," in *Int. Computer Software and Applications Conference*. IEEE, 2009.

[66] S. Huayou, N. Yulin, Y. Lian, and C. Zhong, "A service-oriented analysis and modeling using use case approach," in *Int. Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–6.

[67] Z. Yun, S. Huayou, N. Yulin, and Q. Hengnian, "A service-oriented analysis and design approach based on data flow diagram," in *Int. Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–5.

[68] F. Ricca and A. Marchetto, "A "quick and dirty" meet-in-the-middle approach for migrating toSOA," in *ACM IWPSE-Evol*, 2009.

[69] S-Cube Consortium, "First version of integration framework," S-Cube project, Tech. Rep., 2009.